UNIVERSITY OF ALBERTA

Clustering by Committee

by

Patrick André Pantel ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Department of Computing Science

Edmonton, Alberta
Spring 2003

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisisitons et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-82151-X
Our file Notre référence
ISBN: 0-612-82151-X

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou aturement reproduits sans son autorisation.

Canada

University of Alberta

Library Release Form

Name of Author:	Patrick André Pantel
Title of Thesis:	Clustering by Committee
Degree:	Doctor of Philosophy
Year this Degree Granted:	2003

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

Patrick Pantel

Patrick Pantel
248 Huntington Hill
Edmonton, AB T6H 5Y6
Canada

Date: March 26, 2003

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled *Clustering by Committee* submitted by Patrick André Pantel in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Dr. Dekang Lin
Supervisor

Russell Greiner
Dr. Russell Greiner

Dr. Francis Jeffry Pelletier

Dr. Greg Kondrak

Peter Hooper
Dr. Peter Hooper

Dr. Dan Roth
External Reader

Date: March 26, 2003

pour Carol et mes parents…

Abstract

Text contains a wealth of knowledge about who we are, what we know, how we think, and how we communicate. We are just beginning to tap into the information that is available in the tales we read to our children, the narratives that capture our thoughts, and the stories that shape our world. In this work, we present some recent advances in automatically acquiring knowledge from text. We propose a general-purpose clustering algorithm called CBC (Clustering By Committee) from which we will organize documents according to topics as well as discover concepts and word senses. We will explore the value of these systems by experimenting with two novel evaluation methodologies that attempt to define what a word sense is and define the quality of a particular clustering.

CBC addresses the general goal of clustering, which is to group data elements such that the intra-group similarities are high and the inter-group similarities are low. Using sets of representative elements called committees, CBC attempts to discover cluster centroids that unambiguously describe the members of a possible class. CBC will be shown to outperform several common clustering algorithms in document clustering and concept discovery tasks. Document clustering is practical in many information retrieval tasks such as document browsing and the organization and viewing of retrieval results. Broad-coverage lexical resources such as

WordNet are extremely useful but are mostly hand generated. They often include many rare senses while missing domain-specific senses. Automatically generating them is useful for many applications such as word sense disambiguation, question answering and ontology construction. Sample concepts discovered by CBC include baking ingredients, symptoms, academic departments, Impressionists, Canadian provinces, musical instruments, and emotions.

We present two novel evaluation methodologies. The first is based on the editing distance between output clusters and a manually constructed answer key. It defines how much work is necessary in order to convert from one to the other. For the word sense discovery system, we present an evaluation methodology for measuring the precision and recall of discovered senses. Using WordNet, we formulate what is a correct sense of a word.

Acknowledgments

I have been privileged to work closely with a person I greatly admire, my supervisor Dekang Lin. I am indebted to you for all your support, your guidance, and most of all your friendship. You are an outstanding supervisor and mentor. You have given me opportunity, ethics, and a sound research methodology. I will look back fondly on our late night writing sessions and our long conference calls. Mostly, I will miss just sitting down with you for a late afternoon chat.

I would like to thank my thesis committee members for their valuable comments and constructive criticism; thank you Russ Greiner, Jeff Pelletier, Greg Kondrak, and Peter Hooper. You always seemed to find time for me even with your incredibly busy schedules. I am also grateful to my external reader, Dan Roth, for his assessment and suggestions for future research directions.

Thank you also to the exceptional faculty, staff, and graduate students at the University of Alberta from whom I have gained so much. A special thanks to Randy Goebel, Jonathan Schaeffer, Osmar Zaïane, Edith Drummond, Markian Hlynka, and Daniel Neilson for their guidance, administrative support, many letters of recommendation, and friendship. Thanks also to all the members of the NLP group for their helpful suggestions and stimulating meetings.

Je dois tous mes succès à mes chers parents. Vous avez été des parents models toute ma vie. En grandissant, je n'ai jamais connu autre que l'amour et c'est de là que je trouve ma joie de vivre. Merci des valeurs et attitudes que vous m'avez transmises. Grâce à votre support financier durant mon baccalauréat, j'ai pu me concentrer fort sur mes études et donc j'ai pu recevoir des bourses d'études durant mon doctorat. Mais, encore plus important, merci de votre appui moral

sans lequel il aurait été difficile de compléter les neuf dernières années. Merci aussi à mon frère Christian de tous ses conseils et de m'avoir diverti plusieurs soirs avec nos jeux à l'internet.

Il ne faut pas oublier mes chers grands-parents, oncles, tantes, cousins et cousines que j'apprécie énormément. Un gros merci particulier à tous ceux qui ont fait le voyage à Edmonton pour nous visiter au cours des trois dernières années. Carol et moi attendions toujours avec impatiente vos visites et notre appartement semblait toujours si vide lorsque vous quittiez. Vous serez toujours bienvenus chez nous. Un merci très spécial à Grand-mère et Grand-père. C'est grâce à vous que notre famille maintient des liens si forts. Merci des jeux de 500 quand je me retrouve au Manitoba et des appels hebdomadaires qui me mettent toujours le sourire aux lèvres.

Ceux qui me connaissent savent que je garde toujours le meilleur en dernier. Carol, l'amour de ma vie, tu as été à mes côtés, souvent même quelques pas en avant, depuis le tout début de mes études. Je te remercie de ton appui et encouragement, mais surtout de ton amour. Sans toi, ma vie à l'extérieur du travail n'aurait pas été si belle. Même durant les moments difficiles, où j'étais rarement à la maison en train de travailler sur ma recherche, tu étais toujours pour moi une source d'inspiration et de persévérance. Ton goût de l'aventure rend la vie excitante. Peu importe où nous nous retrouverons l'an prochain, j'anticipe avec joie notre cheminement vers l'avenir. Je partage cette thèse avec toi.

Contents

1 INTRODUCTION **1**

1.1 SUMMARY OF CONTRIBUTIONS 2
1.2 CLUSTERING COMPONENTS 3
  1.2.1 Pattern representation *3*
  1.2.2 Proximity measurements *4*
  1.2.3 Grouping elements *11*
  1.2.4 Data abstraction *11*
  1.2.5 Output interpretation *11*
1.3 CLASSIFICATION VS. CLUSTERING 12
1.4 CLUSTERING ISSUES 12
1.5 OUTLINE 13

2 LITERATURE REVIEW **14**

2.1 HIERARCHICAL ALGORITHMS 15
  2.1.1 AGNES *16*
  2.1.2 DIANA *18*
2.2 PARTITIONAL ALGORITHMS 21
  2.2.1 K-means *22*
  2.2.2 Bisecting K-means *23*
  2.2.3 K-medoids *24*
2.3 HYBRID ALGORITHMS 24
  2.3.1 Buckshot *24*
  2.3.2 BIRCH *25*
  2.3.3 CURE *26*
  2.3.4 Rock *28*
  2.3.5 Chameleon *29*
2.4 OTHER ALGORITHMS 32

3 RESOURCES **33**

3.1 WORDNET 33
3.2 MINIPAR 35
  3.2.1 Parser internals *35*
  3.2.2 Lexicon *39*
  3.2.3 Probability model *40*
3.3 SIMILARITY MEASURE 41

3.3.1 Feature database *41*
3.3.2 Vector-space model *45*
3.3.3 Similarity model *49*

4 CBC **50**

4.1 MOTIVATION 50
4.2 ALGORITHM 52
4.2.1 Phase I *52*
4.2.2 Phase II *54*
4.2.3 Phase III *57*
4.3 COMPARISON WITH UNICON 59
4.4 APPLICATIONS 59
4.4.1 Document clustering *60*
4.4.2 Concept discovery *61*

5 EVALUATION METHODOLOGY **64**

5.1 PREVIOUS APPROACHES 65
5.1.1 Comparison with manually generated answer keys *65*
5.1.2 Application-embedded evaluation *66*
5.2 EDITING DISTANCE 66
5.2.1 Methodology *67*
5.2.2 Consistent extension *70*
5.3 EVALUATING WORD SENSES 71
5.3.1 Precision *71*
5.3.2 Recall *73*
5.3.3 F-measure *73*
5.4 LANGUAGE MODELING 74
5.4.1 Review *74*
5.4.2 Perplexity *75*
5.4.3 Predictive Clustering *77*

6 EXPERIMENTAL RESULTS **78**

6.1 DOCUMENT CLUSTERING 78
6.1.1 Experimental setup *79*
6.1.2 Cluster evaluation *79*
6.1.3 Clustering parameters *80*
6.1.4 Illustrative example *82*
6.2 CONCEPT DISCOVERY 84
6.2.1 Experimental setup *85*
6.2.2 Cluster evaluation *85*
6.2.3 Manual Inspection *86*
6.2.4 Language Modeling *89*
6.2.5 Sample concepts *91*
6.2.6 Word sense discovery *92*

7 CONCLUSIONS **98**

7.1 CONTRIBUTIONS 98
7.2 FUTURE WORK 100

BIBLIOGRAPHY **104**

APPENDIX A **109**

APPENDIX B **114**

APPENDIX C **116**

List of Tables

Table 1.1 – Contingency table for binary variables. 7
Table 1.2 – Example binary feature vectors for college basketball players. 8
Table 3.1 – A subset of the dependency relations in Minipar outputs. 38
Table 6.1 – Document clustering test data sets. 79
Table 6.2 – Cluster quality (%) of various clustering algorithms on document clustering. 80
Table 6.3 – Output of CBC applied to the 46 papers presented at SIGIR-2001 83
Table 6.4 – Concept discovery test data sets. 85
Table 6.5 – Cluster quality (%) of several clustering algorithms on S_{3566} and S_{13403}. 86
Table 6.6 – Manual inspection of five of the 943 clusters discovered by CBC from S_{13403}. 88
Table 6.7 – A comparison of clusters representing the *cell* concept. 89
Table 6.8 – Language modeling training sets. 90
Table 6.9 – Precision, Recall and *F*-measure on S_{13403} for various algorithms. 93
Table 6.10 – Confusion matrix of manual vs. automatic evaluations. 96
Table 7.1 – Top-50 most similar paths to "*X solves Y*" generated by DIRT 102

List of Figures

Figure 1.1 – A clustering example. 2
Figure 2.1 – Taxonomy of clustering algorithms. 15
Figure 2.2 – Dendrogram visualization of a hierarchical clustering result. 16
Figure 2.3 – Clusters discoverable using single-link clustering. 17
Figure 2.4 – The chaining effect in single-link clustering. 17
Figure 2.5 – Single-link vs. complete-link cluster similarity. 18
Figure 2.6 – Single-link, complete-link and average-link clustering. 19
Figure 2.7 – DIANA clustering. 20
Figure 2.8 – *K*-means clustering. 23
Figure 2.9 – Clustering features for two subclusters. 25
Figure 2.10 – Example *CF*-tree. 26
Figure 2.11 – CURE clustering. 27
Figure 2.12 – Chameleon algorithm. 30
Figure 2.13 – Interconnectivity vs. closeness. 31
Figure 3.1 – Example hierarchy of synsets in WordNet. 34
Figure 3.2 – Sample constituency grammar. 36
Figure 3.3 – Example constituency tree. 37
Figure 3.4 – Example dependency tree. 38
Figure 3.5 – Sample entries from Minipar's lexicon. 40
Figure 3.6 – Feature database entry excerpt for the word *Haaretz*. 42
Figure 3.7 – Feature database entry excerpt for the word *umbrella*. 46
Figure 3.8 – Document clustering features using the Bag-of-Words model. 47
Figure 4.1 – Top-20 similar words of *account* and *duty*. 53
Figure 4.2 – Phase II of CBC. 55
Figure 4.3 – Top-30 similar words of *plant*. 58
Figure 4.4 – Top-20 similar words of *wine* and *suit*. 62
Figure 5.1 – Transformation rules example. 68
Figure 5.2 – Counter-example to transformation rules' optimality for soft clusterings. 69
Figure 5.3 – Two examples of sets consistent with classes. 70
Figure 6.1 – CBC evaluation of cluster quality when varying clustering parameters. 82
Figure 6.2 – Cluster quality comparison: document clustering vs. concept discovery. 87
Figure 6.3 – Relative perplexity of predictive clustering on the SJM test set. 91
Figure 6.4 – Relative perplexity of predictive clustering on the AP newswire test set. 92
Figure 6.5 – *F*-measure of several algorithms with $\sigma = 0.18$ and varying θ thresholds. 94
Figure 6.6 – *F*-measure of several algorithms with $\theta = 0.25$ and varying σ thresholds. 95

List of Symbols

n	the number of elements to be clustered	4
m	the dimensionality of the feature space (i.e. the number of all possible features)	4
xij	the j^{th} feature of element x_{ij}, $i \le n$ and $j \le m$	4
xi	the i^{th} element to be clustered, $i \le n$	4
dij	the distance between two elements x_i and x_j	5
K	the number of clusters to be generated by a clustering algorithm	15
DC	the divisive coefficient for the DIANA algorithm	20
T	the number of iterations to run a partitional clustering algorithm	22
α	the number of times *K*-means is applied in the Bisecting *K*-means algorithm	23
N	the *noun* part-of-speech	36
V	the *verb* part-of-speech	36
P	the *preposition* part-of-speech	36
NP	the *noun phrase* constituent	36
VP	the *verb phrase* constituent	36
PP	the *prepositional phrase* constituent	36
X:R:Y	a dependency relationship between elements *X* and *Y* with relationship *R*	43
-X:R:Y	the inverse relationship of *X*:*R*:*Y* (the head and modifier of the relationship are inverted)	45
tf	the term frequency (or Bag-of-Words) vector-space model	46
idf	the inverse document frequency	46
tf-idf	the term frequency – inverse document frequency vector space model	47
mixy	the pointwise mutual information between two events *x* and *y*	47
MIXY	the mutual information between two random variables *X* and *Y*	48

$C(e)$	a frequency count vector for the features of element e	48
$MI(e)$	a mutual information vector for the features of element e	48
$sim(e_i, e_j)$	the similarity between elements e_i and e_j	49
$\theta 1$	the similarity threshold for accepting a candidate committee	54
$\theta 2$	the similarity threshold for determining if an element is covered by a committee	56
$dist(C, A)$	the editing distance between a clustering C and an answer class A	67
C	a set of clusters output by a clustering algorithm	67
A	a set of answer classes (an answer key)	67
B	the baseline clustering where each element is grouped in its own cluster	67
quality	the editing distance-based evaluation methodology	67
$S(w)$	the set of WordNet senses of a word w (each sense is a synset that contains w)	71
θ	the threshold used to determine if a discovered sense of a word is a correct sense	72
$H(X)$	the entropy of a random variable	75
$p(x)$	the probability mass function of a random variable X	75
$H(p, m)$	the cross-entropy of the probability mass functions p and m for a random variable X	75
Reuters	the Reuters-21578 V1.2 document clustering data set	79
20-news	the 20news-18828 document clustering data set	79
S13403	the data set consisting of 13,403 words used in concept discovery	85
S3566	the data set consisting of 3566 words used in concept discovery	85
TREC	1GB newspaper corpus – '88 AP Newswire, '89-'90 LA Times, and '91 SJM	85
$wn(c)$	the WordNet class that has the largest intersection with cluster c	86
Acquaint	3GB newspaper corpus - '96-'00 Xinhua News Service, '98-'00 NYT, '98-'00 AP	91
σ	the similarity threshold for assigning an element to a cluster in a soft clustering	93

Chapter 1

Introduction

Clustering is an exploratory science that discovers patterns in data by grouping data elements according to their similarity. It is a mature field with a rich history spanning many disciplines such as machine learning, natural language processing, image analysis, data mining, information retrieval and bioinformatics. The intuition behind clustering is that elements that belong to a same cluster are similar and elements that belong to different clusters are not similar. For example, in a medical diagnosis system, a clustering algorithm may discover groups of patients with similar diagnoses based on their symptoms.

There are many other applications of clustering. Scouts for professional sports teams may use clustering to find the next superstar. For example, by clustering college basketball players with the likes of known superstars such as Michael Jordan and Kobe Bryant, scouts may identify players who are similar to each superstar. In biotechnology, given a series of micro-array experiments measuring the expression of a set of genes, we can discover groups of genes that vary in similar ways over time. Genes belonging to the same cluster may then be hypothesized as being co-regulated. In data mining, clustering can be used to gain insight on the buying patterns of different groups of people, while in information retrieval, clustering is useful to improve the organization and viewing of retrieval results as well as to generate Yahoo-like hierarchies. Insurance companies may also employ clustering to detect groups of high risk customers.

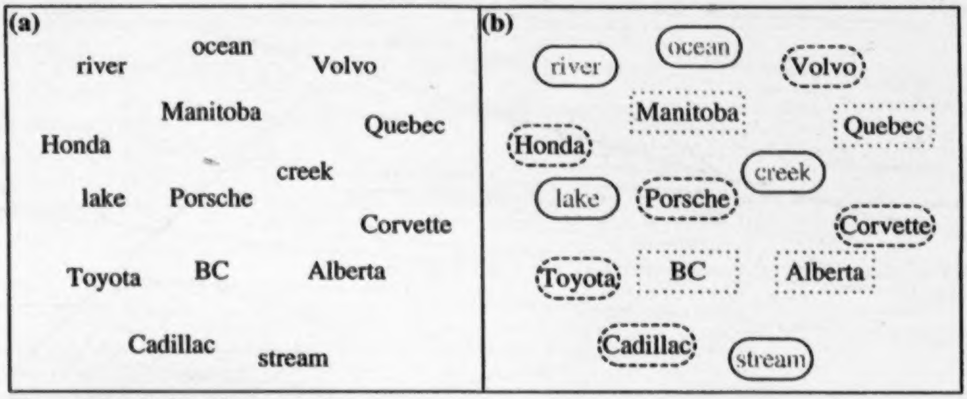

(a)
river
ocean
Volvo
Manitoba
Quebec
Honda
creek
lake
Porsche
Corvette
Toyota
BC
Alberta
Cadillac
stream
(b)
river
ocean
Volvo
Manitoba
Quebec
Honda
creek
lake
Porsche
Corvette
Toyota
BC
Alberta
Cadillac
stream

Figure 1.1 – A clustering example.
(a) a set of English words; (b) a possible clustering of the words (words with the same border belong to the same cluster).

Figure 1.1 shows an example of clustering words according to their meanings. The general goal of clustering is to group data elements such that the intra-group similarities are high (i.e. those elements within a same cluster are very similar) and the inter-group similarities are low (i.e. those elements that are in different clusters are not very similar). The clusters should also span the total set of elements.

1.1 Summary of contributions

This dissertation presents a novel general-purpose clustering algorithm called Clustering by Committee, or CBC for short. It addresses the general goal of clustering, which is to group data elements such that the intra-group similarities are high and the inter-group similarities are low. Using sets of representative elements called committees, CBC attempts to discover cluster centroids that unambiguously describe the members of a possible class.

Although CBC is a general-purpose algorithm, it was designed to learn from textual data. One application that will be presented is document clustering. The goal of document clustering is to discover documents with similar topics. It is practical in many information retrieval tasks such as document browsing and the organization and viewing of retrieval results. CBC will also be applied to the task of automatically discovering concepts by clustering words in large collections of text. Concept discovery is the process of inducing semantic classes such as *countries*, *pastries*, *scientific disciplines* and *sports teams*. Broad-coverage lexical resources such as WordNet are extremely useful but are mostly hand generated. They often include many rare senses while missing domain-specific senses. Automatically generating them is useful for many applications such as word sense disambiguation, question answering and ontology construction. As an extension to concept discovery, CBC will automatically discover the senses of words. CBC will be shown to outperform several common clustering strategies in both document clustering and concept discovery tasks.

Evaluating clustering results has remained a very challenging task. We present a novel evaluation methodology to automatically evaluate clustering output, whose measure is more intuitive and easier to interpret than previous measures. It is based on determining the percentage of savings by using the clustering result to construct an answer key versus constructing it from scratch (i.e. a baseline clustering). We measure this by determining the editing distance, which is the number of operations required to transform a clustering into an answer key.

1.2 Clustering components

Generally, a clustering algorithm consists of the following components (Jain and Dubes 1988):

1) representing the data elements by patterns;
2) defining a proximity measure between patterns;
3) grouping elements according to the proximity measure;
4) abstracting the data; and
5) interpreting the output.

In this section, we describe each of these components and explore common practices used in clustering. We will present the concept of a feature and describe similarity measures for different types of features. The two major categories of clustering algorithms, hierarchical and partitional, will also be introduced.

1.2.1 Pattern representation

Pattern representation typically consists of defining a vector of measurements, called **features**, which will represent each element to be clustered. Features may be of various types such as numerical, binary and categorical. For example, in a medical diagnosis system, patients may be represented by features such as their age (numerical), gender (categorical), blood pressure (numerical) and smoking habits (binary). The set of features used to represent an element makes up the element's **feature vector**. Suppose we are to cluster a set of n elements, $\{x_1, x_2, \ldots, x_n\}$, and each element's feature vector consists of m measurements. The feature vectors are represented in a matrix:

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix}$$

where x_{ij} is the j^{th} feature of element x_i, $i \leq n$ and $j \leq m$.

Some features are more useful than others. Consider the task of clustering text documents according to their topics. Suppose the feature vector for a document consists of the frequency count of all possible words that may be contained in the document. Many words will never occur in a particular document causing many of these features to have zero frequency. **Feature selection** is the process of selecting the most informative features. In this example, we might select only the words that occurred at least k times in the document. Reducing the set of features typically leads to a better clustering. Also, it is useful in making a clustering algorithm more efficient in both time and space complexity.

Word frequencies may not be the best features for clustering documents. The word *the* has a very high frequency in most documents but is not very indicative of the document's topic. Once we have a feature vector of word frequencies, we can define a new feature for each word. Instead of directly using a word's frequency, we can normalize it by the number of documents in which it occurs. Since *the* occurs in most documents, its value will be reduced dramatically. This process of extracting more salient features from a feature vector is called **feature extraction**. Feature selection / extraction may optionally be performed during pattern representation.

1.2.2 Proximity measurements

A **proximity measure** quantifies the closeness between two elements' feature vectors. We measure the proximity using either a **similarity** or **distance** measure depending on the pattern representation. Similarity and distance are essentially synonymous and can usually be interchanged (i.e. the distance between elements measures their dissimilarity).

As a preprocessing step, given a proximity measure, we can convert the $n \times m$ matrix representation of feature vectors into an $n \times n$ half-matrix of the proximity between each element:

$$\begin{bmatrix} 0 & & & & \\ d_{21} & 0 & & & \\ d_{31} & d_{32} & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ d_{n1} & d_{n2} & \cdots & d_{nn-1} & 0 \end{bmatrix}$$

where d_{ij} is the distance or dissimilarity between elements x_i and x_j. Note that the distance between two identical elements is 0 (i.e. the diagonal of the matrix) and that the distances are symmetrical (i.e. $d_{ij} = d_{ji}$). If the proximity metric is not symmetric then the full matrix must be used. This matrix is a general representation of the proximity between elements and it can be used directly in any clustering algorithm.

Since clustering decisions are based on the proximity between feature vectors, the selection of a good proximity measure is critical for the success of a clustering algorithm. For this reason, there are many proximity measures available. The selection of a measure is partially driven by the types of features in the feature vectors. Below we discuss common proximity measures for different feature types.

Numerical features

Numerical features can be discrete or continuous. Suppose we wish to cluster college basketball players and we represent each player by a set of predefined attributes (features). Examples of discrete features include a player's age and jersey number[1]. A player's weight, average points per game and average rebounds per game are examples of continuous features.

Given two elements x_i and x_j represented by numerical feature vectors, the most common distance metrics are derived from the Minkowski metric (Han and Kamber 2001):

---

[1] A player's jersey number can be a minor indicator of his or her skill level since popular numbers are often awarded to better players (e.g. #33 in basketball and #10 in soccer).

$$d_{ij} = \sqrt[p]{\sum_{k=1}^{m} \left| x_{ik} - x_{jk} \right|^{p}} \qquad \text{(Eq. 1.1)}$$

The most common metric, the Euclidean distance, sets $p = 2$:

$$d_{ij} = \sqrt{\sum_{k=1}^{m} \left| x_{ik} - x_{jk} \right|^{2}} \qquad \text{(Eq. 1.2)}$$

Another popular metric, the Manhattan distance, sets $p = 1$:

$$d_{ij} = \sum_{k=1}^{m} \left| x_{ik} - x_{jk} \right| \qquad \text{(Eq. 1.3)}$$

The Minkowski metric is not very reliable for features of different scale (e.g. a player's height and average points per game). The largest-scaled features tend to dominate the other features. A solution is to standardize each feature x_{ij} to unitless measures z_{ij} as follows (Han and Kamber 2001):

1) Compute the mean absolute deviation s_j:

$$s_j = \frac{\sum_{i=1}^{n} \left| x_{ij} - m_j \right|}{n} \qquad \text{(Eq. 1.4)}$$

where m_j is the mean of feature j:

$$m_j = \frac{\sum_{i=1}^{n} x_{ij}}{n} \qquad \text{(Eq. 1.5)}$$

2) Compute the unitless measurement of x_{ij}:

$$z_{ij} = \frac{x_{ij} - m_j}{s_j} \qquad \text{(Eq. 1.6)}$$

Table 1.1 – Contingency table for binary variables.

		x_i	
		1	0
x_j	1	*A*	*B*
	0	*C*	*D*

Other measures of s_j may be used. For example, the standard deviation may be used, however, it is more susceptible to outliers.

Certain features may be more important than others. In our college basketball example, a player's average points per game might be more indicative of his or her ability than his or her shoe size. The weighted Minkowski distance may then be used:

$$d_{ij} = \sqrt[p]{\sum_{k=1}^{m} w_k \left| x_{ik} - x_{jk} \right|^p} \qquad \text{(Eq. 1.7)}$$

where w_k is a predetermined weight of perceived importance of feature k.

The Minkowski metric is also not reliable for features that do not have linear scale (e.g. a feature that measures the growth of a bacteria population). A simple solution is to apply a logarithmic transformation to such features x_{ij}: $z_{ij} = log(x_{ij})$.

Binary features

A binary feature takes on one of two values. We will assume that these values are {0, 1} where a 0 indicates *false*, *no* or the *first choice* and 1 indicates *true*, *yes* or the *second choice*. Consider a prenatal diagnosis system. The gender of the fetus is one possible binary feature where 0 indicates a girl and 1 indicates a boy.

Suppose each element (e.g. a fetus) is represented by m binary features. Table 1.1 gives a contingency table for two elements x_i and x_j. A is the number of features where both x_i and x_j instantiate to 1, B is the number of features where $x_i = 1$ and $x_j = 0$, C is the number of features where $x_i = 0$ and $x_j = 1$ and D is the number of features where both x_i and x_j equal 0. One common measure of similarity between x_i and x_j is the matching coefficient:

Table 1.2 – Example binary feature vectors for college basketball players.

Element	*NAME*	*ATTENTED IV-LEAGUE SCHOOL*	*ALL-AMERICAN*	*HONOR ROLL*	*SHOOTS RIGHT-HANDED*	*AMERICAN CITIZEN*
1	Roland	N	Y	Y	N	N
2	Ed	Y	Y	N	Y	N
3	Chris	Y	N	Y	Y	Y

$$d_{ij} = \frac{A+D}{A+B+C+D} \qquad \text{(Eq. 1.8)}$$

This measure works well for binary features like the *gender* feature in the prenatal example since either value attached to the feature is equally weighted. Such features are called symmetric. Asymmetric features must be treated differently. An example of an asymmetric feature is "Positive Mother's Alphafetoprotein Screening Blood Test". Most fetuses with spina bifida will cause a positive test. However, two fetuses are not inherently similar if they both cause a negative test. For asymmetric features, the Jaccard coefficient (Jain, Murty and Flynn 1999) is a better measure:

$$d_{ij} = \frac{A}{A+B+C} \qquad \text{(Eq. 1.9)}$$

Consider the binary feature vectors of three college basketball players in Table 1.2. The matching coefficient measures are:

$$d_{12} = \frac{1+1}{1+1+2+1} = 0.4$$

$$d_{13} = \frac{1+0}{1+1+3+0} = 0.2$$

$$d_{23} = \frac{2+0}{2+1+2+0} = 0.4$$

and the Jaccard coefficient measures are:

$$d_{12} = \frac{1}{1+1+2} = 0.25$$

$$d_{13} = \frac{1}{1+1+3} = 0.2$$

$$d_{23} = \frac{2}{2+1+2} = 0.4$$

Categorical features

Categorical features are a generalization of binary features. The difference is that the domain of a categorical feature is not limited to {0, 1}. Instead, it could be any one of a finite set of values. There are two types of categorical features: nominal and ordinal. In nominal features, the actual ordering of values is meaningless. For example, a basketball player's eye color may instantiate to one of {*black*, *blue*, *brown*, *green*, *hazel*} but there is no notion of order. A common similarity measure between two elements x_i and x_j whose features are all nominal is the matching coefficient:

$$d_{ij} = \frac{A}{A+B} \qquad \text{(Eq. 1.10)}$$

where A is the total number of features in agreement between x_i and x_j while B is the total number of disagreements between x_i and x_j.

Another way of computing the similarity is to transform each nominal feature into a set of binary features and then apply the similarity metrics described above in the *Binary features* section. A nominal feature that instantiates to t values is transformed into t binary variables, each representing the truth of the value. For example, the eye color feature from above is transformed into five binary features: *eye color black*, *eye color blue*, etc.

An ordinal feature is like a nominal feature except that the ordering of its values is meaningful. For example, a person's highest conferred degree/diploma may be a feature that takes on the following values {*high school*, *certificate*, *diploma*, *Bachelor*, *Masters*, *Ph.D.*}. We can represent these values by their relative ranking (e.g. {1, 2, 3, 4, 5, 6} for the previous example). We then compute the similarity between two elements whose features are all ordinal using the methods from the *Numerical features* Section. Since most ordinal features will have varying sizes of domains, we can normalize all relative rankings in the range [0, 1].

Combining features of mixed types

Most applications will not have homogeneous feature types. There are two common ways of dealing with feature vectors of mixed types. The first simply groups together features of the same type and computes the similarity for each separate group. The resulting set of similarities must then be comparable in order to obtain an overall similarity.

Another method computes the similarity between two elements of mixed type feature vectors using all features simultaneously (Han and Kamber 2001):

$$d_{ij} = \frac{\sum_{k=1}^{m} \delta_{ij}^{k} d_{ij}^{k}}{\sum_{k=1}^{m} \delta_{ij}^{k}} \qquad \text{(Eq. 1.11)}$$

where $\delta_{ij}^{k} = 0$ if (i) x_{ik} or x_{jk} has no measurement or (ii) $x_{ik} = x_{jk} = 0$ and feature k is asymmetric binary; otherwise, $\delta_{ij}^{k} = 1$. d_{ij}^{k} is the contribution of feature k on the distance measurement between elements x_{ik} and x_{jk}. It is computed differently depending on feature k's type:

- k is binary or nominal: $d_{ij}^{k} = 0$ if $x_{ik} = x_{jk}$; otherwise $d_{ij}^{k} = 1$
- k is numerical: $d_{ij}^{k} = \frac{|x_{ik} - x_{jk}|}{\max_h x_{hk} - \min_h x_{hk}}$, where h can be any element x_h such that x_{hk} has a measurement

Note that if k is ordinal, it can be converted to numerical as described in the above *Categorical features* Section.

1.2.3 Grouping elements

The grouping component is what is often referred to as the **clustering algorithm**. Generally, clustering algorithms are classified as hierarchical vs. partitional. Hierarchical algorithms use a proximity measure to create a hierarchical decomposition of the data elements. Clusters are iteratively merged or split until a stopping condition is met. Partitional algorithms produce a single partitioning by optimizing some criterion. The next chapter provides a survey of commonly used clustering algorithms.

1.2.4 Data abstraction

Data abstraction, an optional component of the clustering process, is the practice of compacting the representation of clusters. This may be required for making the clustering more efficient or for making the output clusters interpretable by humans. Many clustering algorithms represent a cluster by a set of representative elements (often just one) or by an artificial element (often constructed by averaging the feature vectors of the elements of the cluster). This representation allows more efficient similarity computations between clusters since pairwise comparisons of elements are not required.

1.2.5 Output interpretation

Finally, **output interpretation** is the optional procedure of using or evaluating the clusters discovered by the clustering process. Evaluating clustering results is a very difficult task. Evaluation methodologies generally fall under two categories:

- comparing cluster outputs with manually generated answer keys; and
- embedding the clusters in an application and using its evaluation measure.

The methods corresponding to the first category typically measure a specific property of the clusters. However, these properties are not directly related to application-level goals of clustering. The second category is goal-oriented however these methods generally do not apply to clustering algorithms that are not designed for the application. Examples of each category are described in Section 5.1.

1.3 Classification vs. clustering

It is important to make the distinction between classification and clustering. Classification is a supervised learning problem where the target classes are predefined. The training examples consist of a series of data elements with their target class (usually manually assigned) and the learning algorithm typically uses these elements to discover a description of the classes. The goal of classification is to place a new element into the class whose description best fits the element. An example of classification is to classify electronic mail messages into one of two categories *spam* and *not_spam* (Pantel and Lin 1998). Here, a collection of manually tagged spam and non-spam electronic mail is collected and is used to describe the two classes using a naïve Bayes model. In contrast to classification, clustering is an unsupervised learning problem. The classes are not designated a priori. Instead, the algorithm searches for natural groupings of elements according to their similarity. Also, the number of clusters is unknown.

1.4 Clustering issues

Choosing a proper clustering strategy for a particular application is driven by the type of data elements and the nature of the application. Consequently, many clustering strategies exist only to deal with the particular intricacies of a single application. Following are some general issues regarding clustering systems.

- **Scalability**: Many algorithms, such as the hierarchical clustering algorithms described in the next chapter, cannot easily handle a large number of data elements in high-dimensional space. A clustering algorithm must have efficient space and running time complexity to be able to deal with such data.

- **Ability to deal with features of different types**: In many applications, features are of varying types. Computing the similarity between two elements must combine the different feature types in some way.

- **Ability to classify elements in sparse feature-spaces**: High dimensional data is very challenging because it is often very sparse and skewed. Clustering algorithms should be able to handle such data sets.

- **Ability to classify unknown elements into existing clusters**: A clustering algorithm should be able to assign unseen elements into existing clusters in an online fashion.

- **Ability to classify elements with few outside parameters requiring domain knowledge**: Many clustering algorithms require input parameters from the user. For example, in most partitional clustering algorithms, the output number of clusters must be predetermined. Outside information constrains the discovery process and should be minimized.

- **Ability to deal with noisy data**: Data sets are often not clean. They contain erroneous elements and outliers. Clustering algorithms must be robust enough to minimize the effect of such elements.

1.5 Outline

Following is a brief description of the chapters in this dissertation:

- **Chapter 1 – Introduction**: Motivations for clustering, an introduction to the contributions of this dissertation, and a description of common clustering properties and practices.

- **Chapter 2 – Literature Review**: A survey of commonly used clustering algorithms.

- **Chapter 3 – Resources**: A definition of the terms and notation used in the dissertation as well as a description of the required resources for our work (including WordNet, a parser called Minipar, a feature database, a vector-space model, and a similarity model).

- **Chapter 4 – CBC**: A description of CBC as a general-purpose clustering algorithm as well as its application to document clustering and concept discovery.

- **Chapter 5 – Evaluation Methodology**: A description of the methodology used to evaluate CBC, including two novel evaluation methodologies.

- **Chapter 6 – Experimental Results**: The experimental results obtained by applying the evaluation methodology of Chapter 5.

- **Chapter 7 – Conclusions**: A synthesis of the contributions of this dissertation as well as future applications of CBC.

Chapter 2

Literature Review

Clustering algorithms are generally categorized as partitional and hierarchical. This chapter describes some common clustering algorithms, which are shown in the taxonomy of Figure 2.1. The shaded algorithms are only briefly discussed. Here are general properties that characterize clustering algorithms (Jain, Murty, Flynn 1999):

- **Agglomerative vs. Divisive**: In agglomerative algorithms (bottom-up approach), each element is initially its own cluster and then the most similar clusters are iteratively merged until we are left with one large cluster containing all elements or until a stopping condition is met. Conversely, divisive algorithms (top-down approach) initially begin with a single all-encompassing cluster and iteratively split the clusters until each element belongs to its own cluster or until a stopping condition is met.
- **Hard vs. Soft**: Hard clustering algorithms assign each element to exactly one cluster whereas soft (fuzzy) algorithms may assign an element to multiple clusters. In soft clustering, a **membership degree** is associated with each element's assignment to a cluster.
- **Deterministic vs. Stochastic**: These types of searches mostly apply to partitional algorithms that optimize some clustering function. Stochastic algorithms use random searches of the feature space while deterministic algorithms do not.

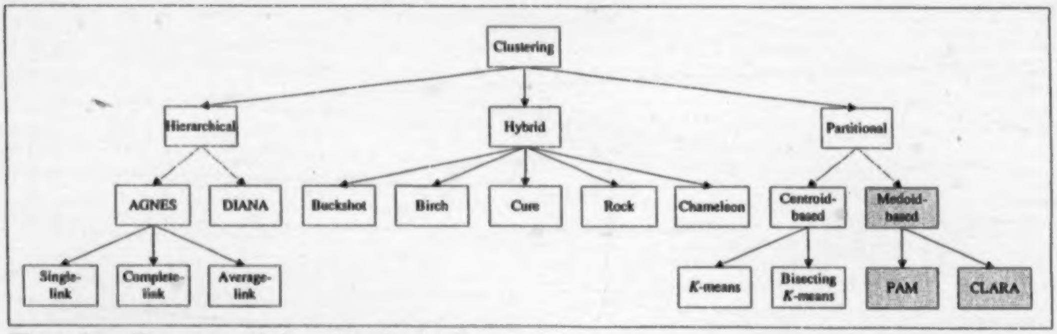

Clustering
Hierarchical
Hybrid
Partitional
AGNES
DIANA
Buckshot
Birch
Cure
Rock
Chameleon
Centroid-based
Medoid-based
Single-link
Complete-link
Average-link
K-means
Bisecting K-means
PAM
CLARA

Figure 2.1 – Taxonomy of clustering algorithms.
The algorithms within the taxonomy will be presented in this chapter. Shaded algorithms are only briefly discussed.

Throughout this chapter, we use *n* to represent the number of elements that are to be clustered. When the number of clusters must be fixed by an input parameter, like in many partitional clustering algorithms, we refer to this number by *K*.

2.1 Hierarchical algorithms

Hierarchical algorithms produce a nested partitioning of the data elements by merging or splitting clusters. Agglomerative algorithms iteratively merge clusters until an all-encompassing cluster is formed while divisive algorithms iteratively split clusters until each element belongs to its own cluster. The merge and split decisions are based on the similarity metric. The resulting decomposition (tree of clusters) is called a **dendrogram**.

Figure 2.2 shows a possible dendrogram produced by an agglomerative hierarchical algorithm. At the topmost level of the dendrogram, we have a single cluster containing all elements. Using a similarity threshold, we can extract a clustering of the data by cutting the dendrogram according to this threshold. Then, each connected component of the dendrogram forms a cluster. For example, assuming that the best clustering in the 2-dimensional space of Figure 2.2 consists of small tight clusters, the dotted line in (b) gives a good threshold for this data resulting in three clusters: {A, E, C}, {H, I} and {D, B, G, F}. The problem with any threshold is that on some data sets, a particular threshold will be good but on another data set, it will fail. For example, in Figure 2.2, if the similarity threshold was just a little higher, we would have five clusters with elements *C* and *D* in separate clusters.

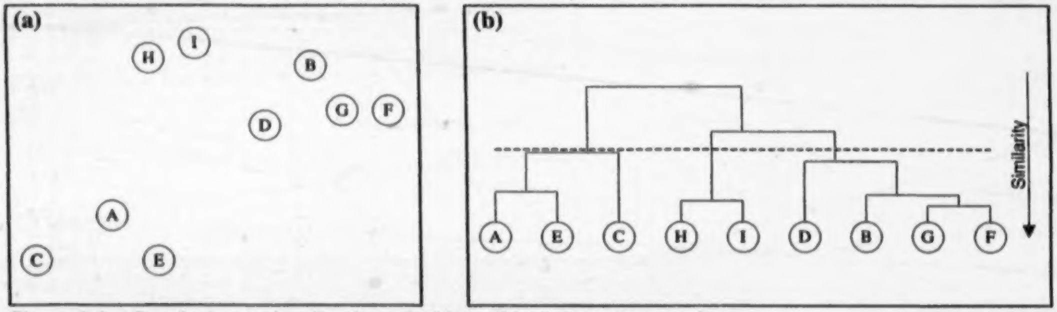


Figure 2.2 – Dendrogram visualization of a hierarchical clustering result.
(a) Nine data points in 2-dimensional space; (b) the dendrogram produced by a hierarchical agglomerative clustering algorithm (the dotted line indicates a possible similarity threshold for selecting the final clustering).

The dendrogram provides a visualization of how the algorithm produced its output. For example, if a particular output cluster is bad, the dendrogram provides a method of verifying how this bad cluster was formed. Hierarchical algorithms rigidly make merge and split decisions. If a particular decision is wrong, the algorithm will never go back and undo the decision. This makes the algorithm more efficient than performing a combinatorial search of all possible decisions but it can never correct itself.

2.1.1 AGNES

AGNES (AGglomerative NESting) is a standard agglomerative clustering algorithm (Kaufmann and Rousseeuw 1990):

1) initially start with *n* clusters each containing a different element;

2) merge the two most similar clusters (repeat $n - 1$ times).

In the final step, an all-encompassing cluster is created and the result is a dendrogram like the one in Figure 2.2. The different versions of AGNES differ in how they compute cluster similarity. The most common versions of AGNES are single-link, complete-link and average-link clustering. The complexity of these algorithms is $O(n^2 \log n)$ (Jain, Murty, Flynn 1999).

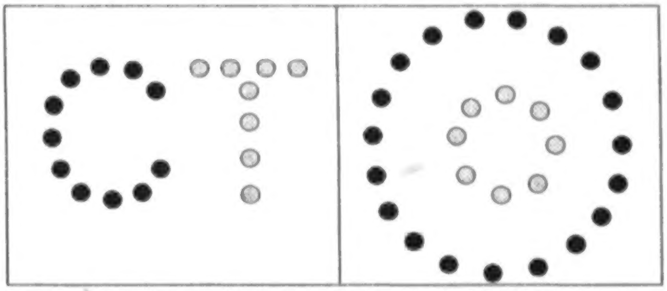


Figure 2.3 – Clusters discoverable using single-link clustering. Complete-link and average-link cannot discover these two clusterings.

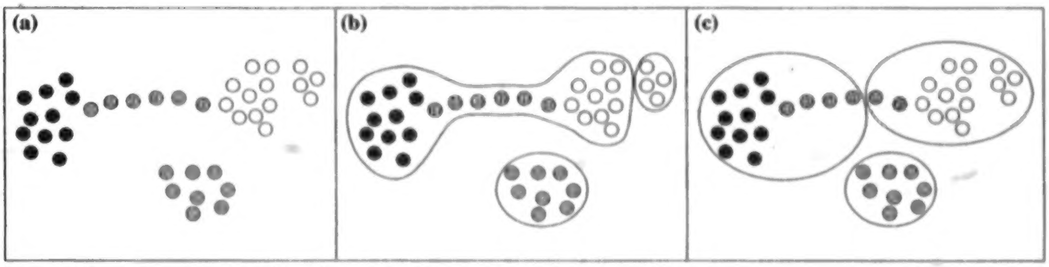


Figure 2.4 – The chaining effect in single-link clustering.
(a) Data points in 2-dimentional space; (b) the clustering produced by single-link clustering; (c) the clustering produced by complete-link clustering. The proximity measure is the Euclidean distance.

Single-link clustering

In single-link clustering (Sneath and Sokal 1973), the similarity between two clusters is the similarity between their most similar members (e.g. using the Euclidean distance). It is capable of discovering clusters of varying shapes like the clusters of Figure 2.3. However, single-link is not practical because it suffers from the chaining effect (Nagy 1968). For example, in Figure 2.4 (b), single-link clustering generates an elongated cluster because of a bridge of elements connecting two clusters.

Complete-link clustering

In complete-link clustering (King 1967), the similarity between two clusters is the similarity between their least similar members (e.g. using the Euclidean distance). Although complete-link clustering is not capable of discovering clusters like the two in Figure 2.3, it does not suffer from the chaining effect. Rather than producing straggly elongated clusters like single-link, complete-

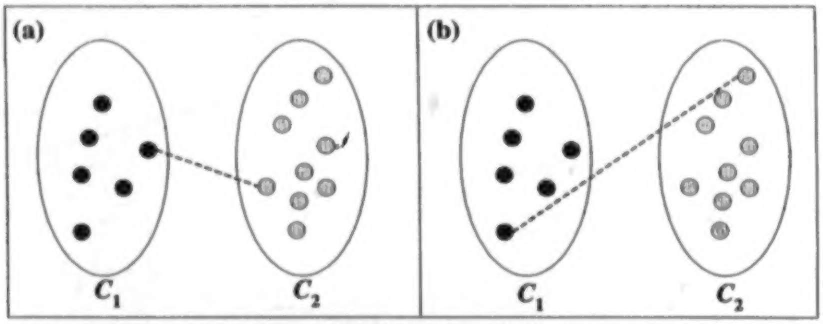


Figure 2.5 – Single-link vs. complete-link cluster similarity.
C_1 and C_2 are two clusters in 2-dimensional space where their similarity is the similarity between the two elements joined by a dotted line for (a) the single-link algorithm and (b) the complete-link algorithm.

link generates compact clusters (Baeza-Yates 1992). Figure 2.4 (c) shows an example. Complete-link generates better clusterings than single-link in many applications (Jain and Dubes 1988). Figure 2.5 illustrates the different computations for cluster similarity between single-link and complete-link.

Average-link clustering

Average-link clustering (Han and Kamber 2001) produces similar clusters to complete-link clustering except that it is less susceptible to outliers. It computes the similarity between two clusters as the average similarity between all pairs of elements across clusters (e.g. using the Euclidean distance). Figure 2.6 shows snapshots of merge decisions comparing the three linkage algorithms on a 2-dimensional data set.

2.1.2 DIANA

DIANA (DIvisive ANAlysis) is a standard divisive clustering algorithm (Kaufmann and Rousseeuw 1990) although it is not as common as AGNES. Divisive clustering algorithms start with a single cluster containing all elements. Considering all possible splits of the cluster into two clusters gives $2^{n-1} - 1$ possibilities. Using a splitting heuristic to iteratively split the largest cluster, DIANA has worst-case time complexity $O(n^2 \log n)$.

Let the diameter of a cluster *c* be the similarity between the two least similar elements in *c*. The algorithm is as follows:

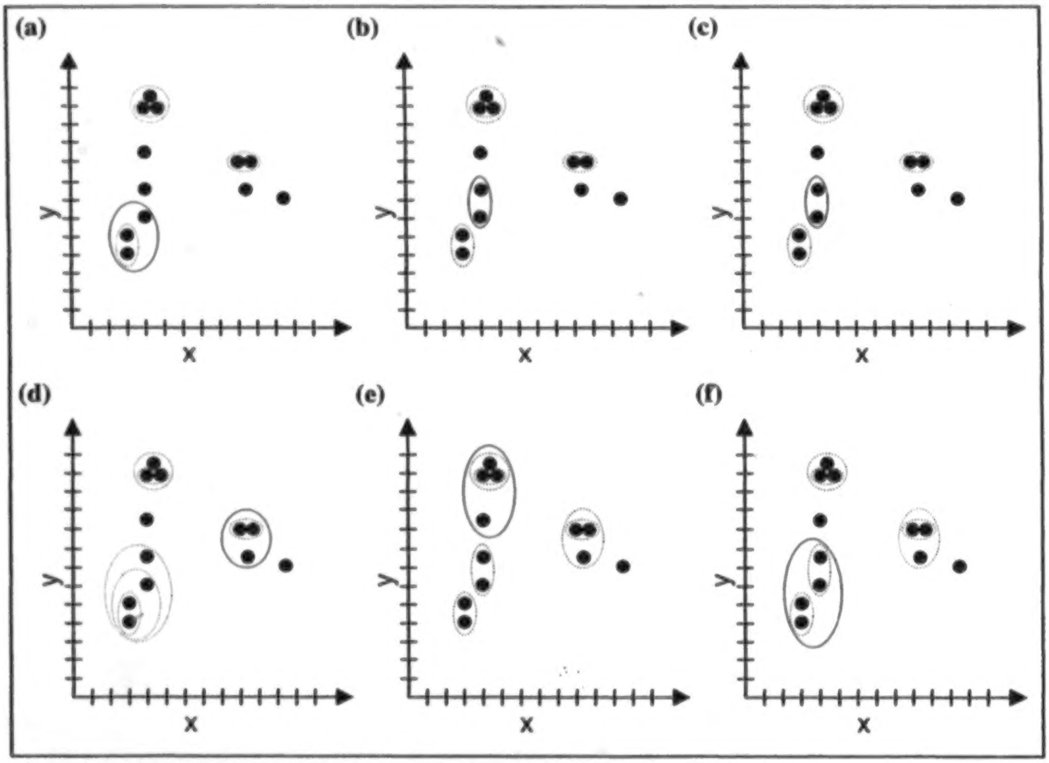


Figure 2.6 – Single-link, complete-link and average-link clustering.
Dotted ellipses denote previously merged clusters and solid ellipses denote newly merged clusters. (a), (b) and (c) illustrate the fifth merge decisions for single-link, complete-link and average-link respectively while (d), (e) and (f) illustrate the seventh merge decisions.

1) initially start with a single cluster encompassing all elements;
2) select *l*, the largest cluster or the cluster with highest diameter;
3) find the element *e* in *l* that has the lowest average similarity to the other elements in *l*;
4) *e* is the first element added to the splinter group while the other elements in *l* remain in the original group;
5) find the element *f* in the original group that has highest average similarity with the splinter group;
6) if the average similarity of *f* with the splinter group is higher than its average similarity with the original group then assign *f* to the splinter group and go to Step 5; otherwise do nothing;
7) repeat Step 2 – Step 6 until each element belongs to its own cluster.

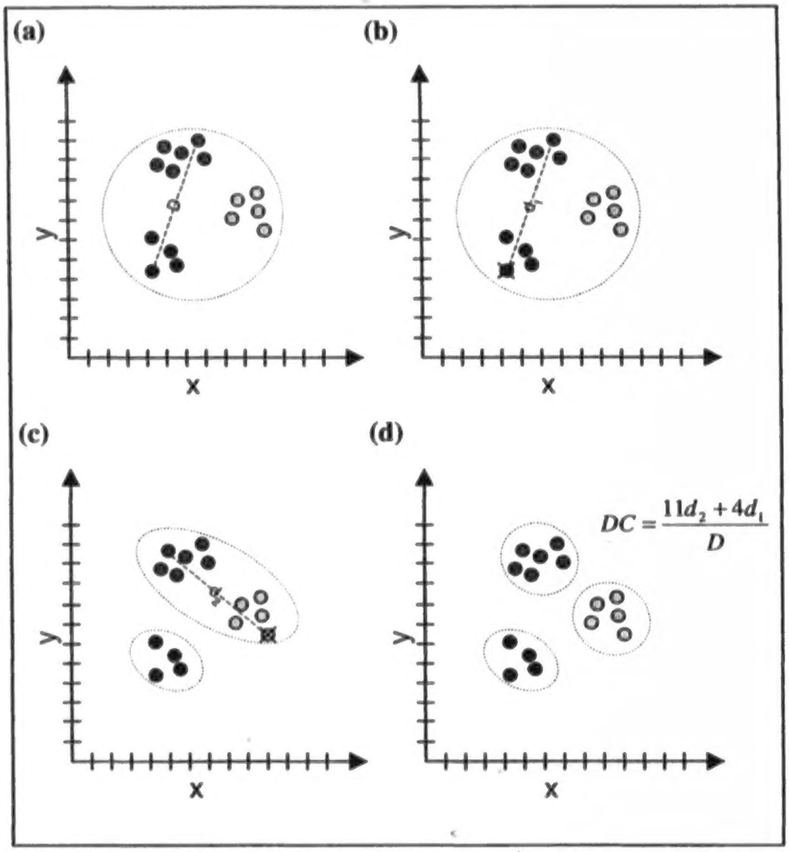


Figure 2.7 – DIANA clustering.
(a) the initial all-encompassing cluster with diameter D; (b) the first splinter group defined by the cross (d_1 is the diameter of l from Step 2); (c) the result of the reassignment of elements to the splinter group after the first iteration - the new splinter group is defined by the cross (d_2 is the diameter of the new l from Step 2); (d) the result of the reassignment of elements to the splinter group after the second iteration and the DC measure assuming that this is the final partitioning.

After completion, each element will belong in its own cluster (i.e. there will be n clusters). DIANA provides a measure of the strength of the clustering structure called the divisive coefficient, DC:

$$DC = \frac{\left(\sum_{e \in D} d(e)\right)}{d} \qquad \text{(Eq. 2.1)}$$

where *D* is the set containing all elements to be clustered, *d(e)* is the diameter of the last cluster to which element *e* belonged before being split to a single-element cluster, and *d* is the diameter of *D*. The higher the *DC*, the stronger becomes the clustering structure. *DC* will be lowest when each element's before-last-split results in a very tight cluster. However, the union of before-last-splits of the elements is rarely the desired clustering. When using the hierarchy given by a hierarchical clustering algorithm, one usually obtains a partitioning by applying a similarity threshold on the hierarchy. Defining *d(e)* as the diameter of the cluster to which element *e* belonged before being split into the cluster in which it resides in the final partitioning (by applying some threshold on the hierarchy) gives a better indication of the strength of the clustering structure. Here, an element lowers the *DC* if its last split before the final partitioning was unnecessary.

Figure 2.7 shows an example of clustering using DIANA. The different shadings represent a possible target clustering. In (a), the all-encompassing cluster is shown as well as the diameter *D* of the data set, which is used in computing the divisive coefficient. The element with the cross in (b) is the element with the lowest average similarity to all other elements and it defines the first splinter group. The small cluster in (c) shows all the elements that were added to the splinter group (Step 5 and Step 6 of the algorithm). The larger cluster is then selected as the next cluster to split since it has the largest diameter, shown by d_2. The element with the cross in (c) represents the next splinter group. The resulting reassignment of elements to the splinter group is shown in (d) as well as the divisive coefficient assuming that this is the selected partitioning.

2.2 Partitional algorithms

Partitional algorithms do not produce a nested series of partitions. Instead, they generate a single partitioning, often of predefined size *K*, by optimizing some criterion. A combinatorial search of all possible clusterings to find the optimal solution is clearly intractable. The algorithms are then typically run multiple times with different starting points. Partitional algorithms are not as versatile as hierarchical algorithms but they often offer more efficient running time (Jain, Murty and Flynn 1999).

2.2.1 *K*-means

The most commonly used family of partitional algorithms is based on the *K*-means algorithm (McQueen 1967). *K*-means clustering is often used on large data sets since its complexity is linear in *n*, the number of elements to be clustered. It creates a partitioning such that the intra-cluster similarity is high and the inter-cluster similarity is low. *K*-means uses the concept of a centroid where a centroid represents the center of a cluster. A centroid is usually not an element from the cluster. Rather, it is a pseudo-element that represents the center of all other elements. Often the mean of the feature vectors of the elements within a cluster is used as that cluster's centroid. It is often difficult to define a centroid for categorical features.

K-means iteratively assigns each element to one of *K* clusters according to the centroid closest to it and recomputes the centroid of each cluster as the average of the cluster's elements. The following steps outline the algorithm for generating a set of *K* clusters:

1) randomly select *K* elements as the initial centroids of the clusters;
2) assign each element to a cluster according to the centroid closest to it;
3) recompute the centroid of each cluster as the average of the cluster's elements;
4) repeat Steps 2-3 for *T* iterations or until a criterion converges, where *T* is a predetermined constant.

The most commonly used criterion is the squared-error criterion, *E*:

$$E = \sum_{i=1}^{K} \sum_{e \in c_i} |e - m_i|^2 \qquad \text{(Eq. 2.2)}$$

where e is an element in cluster c_i and m_i is the centroid of c_i. Figure 2.8 illustrates the operation of *K*-means on 2-dimensional elements with K=4. In the initialization of *K*-means, four elements are chosen as the initial centroids (represented by crosses). After the sixth iteration of the algorithm, shown in (*f*), the element assignments to clusters will no longer change and the algorithm terminates.

K-means has complexity $O(K \times T \times n)$ and is efficient for many clustering tasks since the parameters *K* and *T* are usually small fixed constants. Because the initial centroids are randomly

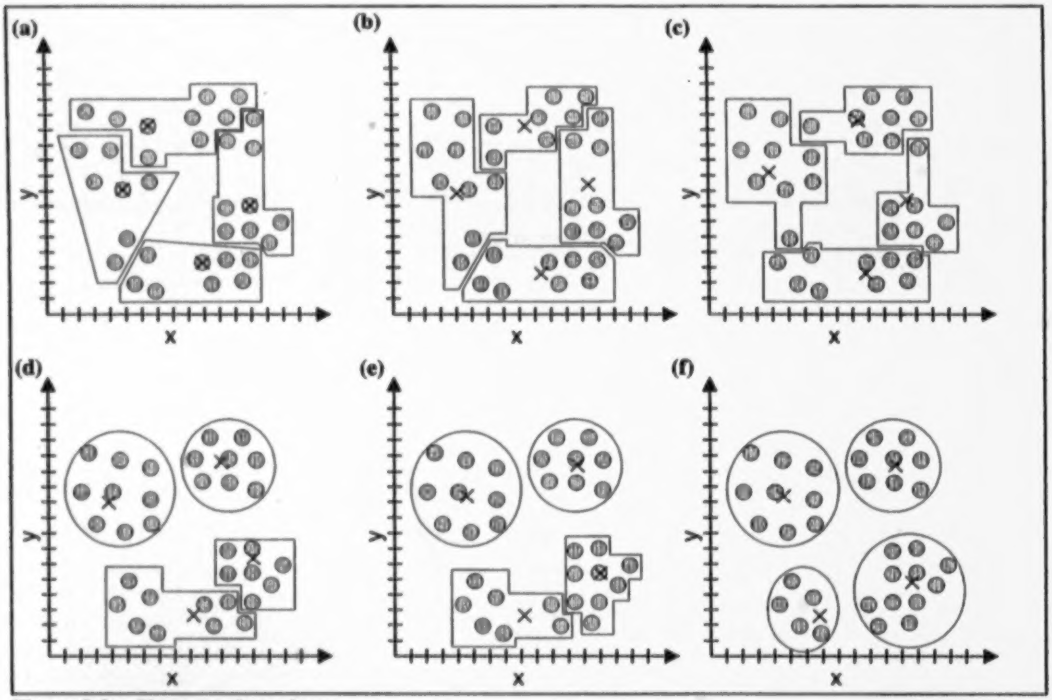


Figure 2.8 – *K*-means clustering.
The crosses represent cluster centroids and K=4. (a) the initial randomly selected centroids and the first cluster assignment; (b) – (f) the second to sixth iterations of *K*-means. After the sixth iteration, the element assignments do not change and the algorithm terminates.

selected, the resulting clusters vary in quality. Some sets of initial centroids lead to poor convergence rates or poor cluster quality.

2.2.2 Bisecting *K*-means

Bisecting *K*-means (Steinbach, Karypis and Kumar 2000), a divisive variation of *K*-means, begins with a set containing one all-encompassing cluster consisting of every element and iteratively picks the largest cluster in the set, splits it into two clusters and replaces it by the split clusters. Splitting a cluster consists of applying the *K*-means algorithm α times with K=2 and keeping the split that has the highest average element-centroid similarity. Note here that $\alpha \neq T$. It is the whole *K*-means algorithm that is repeated α times. Each instantiation of *K*-means will have T iterations.

2.2.3 *K*-medoids

The centroids constructed by *K*-means are sensitive to outliers, if there are many of them, since each element has a direct influence on the construction of the centroids. *K*-medoids (Kaufmann and Rousseeuw 1987) is a family of algorithms that addresses this shortcoming. Instead of representing a cluster by its centroid, *K*-medoids uses one of the elements of the cluster as its representative. The algorithm is very similar to *K*-means. Initially, *K* random elements are chosen as the initial representative of the *K* clusters. In each iteration of the algorithm, a representative element is replaced by a randomly chosen non-representative element if the criterion (e.g. squared-error criterion) is improved. The elements are then reassigned to their closest cluster. Examples of *K*-medoids algorithms include PAM (Kaufmann and Rousseeuw 1987) and CLARA (Kaufmann and Rousseeuw 1990).

2.3 Hybrid algorithms

Hybrid clustering algorithms are characterized as multi-phase algorithms that combine hierarchical and partitional techniques. CBC falls within this class of algorithms. In this section, we present five algorithms: Buckshot, BIRCH, CURE, Rock and Chameleon.

2.3.1 Buckshot

Buckshot (Cutting, Karger, Pedersen and Tukey 1992) addresses the problem of randomly selecting initial centroids in *K*-means by combining it with average-link clustering. Buckshot first applies average-link to a random sample of $\sqrt{n}$ elements to generate *K* clusters. It then uses the centroids of the clusters as the initial *K* centroids of *K*-means clustering.

As the random sample-size approaches *K*, Buckshot degenerates to the *K*-means algorithm. The strict definition of the sample size makes Buckshot unsuitable for some situations. Suppose one wish to cluster 100,000 documents into 1000 newsgroup topics. Buckshot could generate at most $\sqrt{100{,}000} \approx 316$ initial centroids. The sample size counterbalances the quadratic running time of average-link to make Buckshot efficient: $O(K \times T \times n + n\log n)$. However, the algorithm can be run with any sample size as long as the speed of clustering is acceptable.

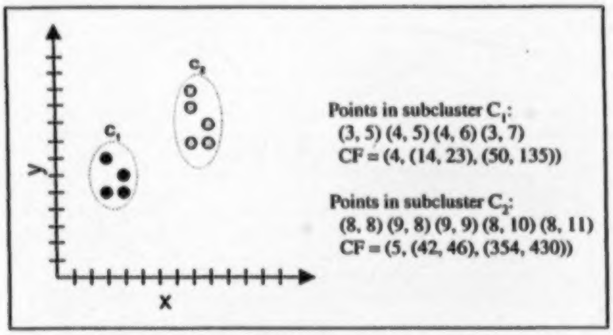


Figure 2.9 – Clustering features for two subclusters.

2.3.2 BIRCH

BIRCH, Balanced Iterative Reducing and Clustering using Hierarchies (Zhang, Ramakrishnan and Livny 1996), is a two phase algorithm that uses a structure called a *CF*-tree to abstract the data yielding an efficient algorithm. A *CF*-tree is a compression of the data elements that attempts to preserve the inherent structure of the data. The two phases are:

1) construct a *CF*-tree by scanning through each element to be clustered;

2) apply any clustering algorithm to cluster the leaf nodes of the *CF*-tree.

A *CF*-tree is a hierarchy of sets of clustering features. Given a subcluster whose elements are represented by *m*-dimensional feature vectors, a clustering feature, *CF*, summarizes the information contained in the elements:

$$CF = \left(N, \overrightarrow{LS}, \overrightarrow{SS}\right) \qquad \text{(Eq. 2.3)}$$

where N is the number of elements in the subcluster, $\overrightarrow{LS} = \sum_{i=1}^{N} \vec{x}_i$ and $\overrightarrow{SS} = \sum_{i=1}^{N} \vec{x}_i^{\,2}$. Figure 2.9 shows an example of two *CF*'s in 2-dimensional feature space.

Figure 2.10 shows an example of a *CF*-tree. A non-leaf node summarizes the statistics of its children by storing the sum of the *CF*'s of its children. A maximum branching factor, *b*, must be given. This is the maximum number of children that a non-leaf node may have. In Figure 2.10, $k \leq b$. Also, the user must specify a maximum diameter threshold *d*. This is the maximum distance allowed between elements of a subcluster at a leaf node.

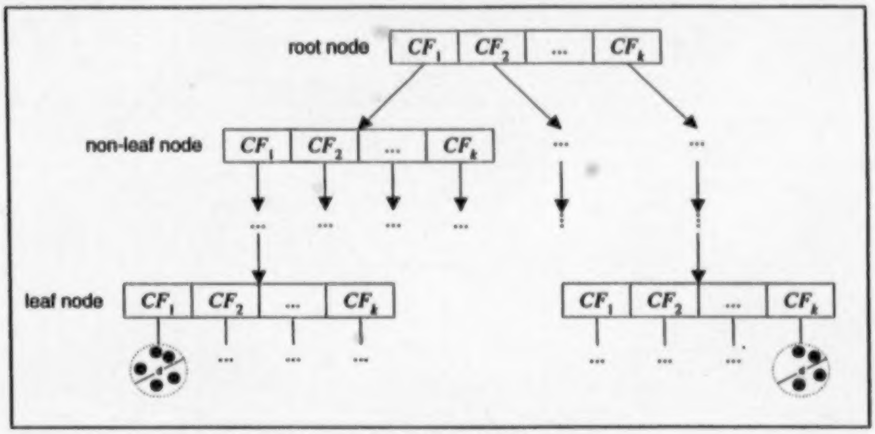


Figure 2.10 – Example *CF*-tree.

In Step 1 of BIRCH, elements are taken one at a time and they are inserted into the subcluster of the leaf node to which it is closest. If an insertion causes the diameter of the subcluster to exceed d, then the leaf node is split into two using a splitting criterion like the one discussed in Section 2.1.2. A split may result in the parent node to be split if its branching factor exceeds b. This process of splitting can propagate all the way up to the root node if all antecedents' branching factors exceed b. After the insertion is completed, the *CF*'s of all antecedent nodes in the tree are updated to maintain the sum of their children's *CF*'s.

The first step of BIRCH has time complexity $O(n)$. As long as the chosen algorithm for step 2 is also linear (e.g. a partitional algorithm like K-means), BIRCH has overall time complexity $O(n)$, which is more efficient than AGNES and DIANA. Because BIRCH uses a diameter parameter, it is not very good for discovering clusters that are not spherical. Another problem with BIRCH is that it is sensitive to the order in which the elements are scanned in Step 1 of the algorithm.

2.3.3 CURE

Single-link clustering has the advantage of being able to discover clusters of various shapes and sizes but it is not robust in the presence of outliers (i.e. the chaining effect). CURE, Clustering Using REpresentatives (Guha, Rastogi and Shim 1998), is similar in operation to single-link clustering but is more robust to outliers. Clusters are represented by a set of initially well-scattered points that are shrunk towards the center of gravity of the cluster.

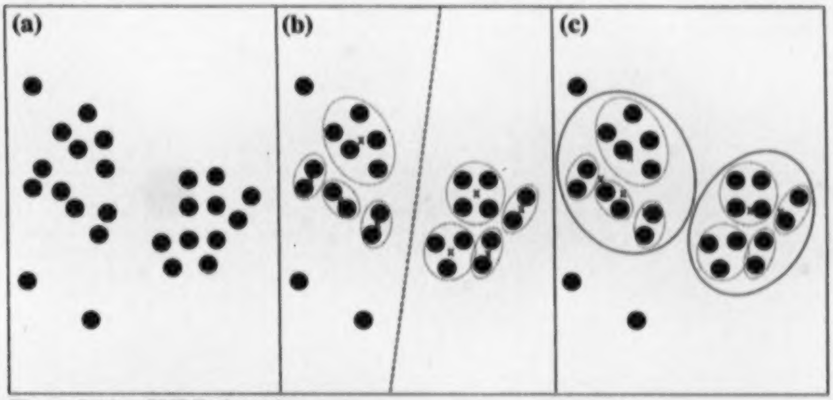


Figure 2.11 – CURE clustering.
(a) a sample set of 2-dimensional elements; (b) the single partition created is represented by the dashed line, the partial clustering is represented by the dotted ellipses and the crosses represent cluster means; (c) the resulting clusters with representative elements shrunk towards the cluster centers.

Given a set of elements *X* to cluster, CURE initially selects a random sample of size *s* from *X*. The random sample is then partitioned into *p* partitions each with size *s*/*p* and then the partitions are partially clustered using an agglomerative hierarchical algorithm (e.g. AGNES). Setting a high similarity threshold in AGNES gives many small clusters. Clusters that grow too slowly are tagged as outliers and are eliminated. At this point, we have several small tight clusters and each is represented by the mean of its constituting elements (a centroid as described in Section 2.2.1). Figure 2.11 (a) shows a set of 2-dimensional elements to be clustered using CURE while (b) shows the partition, the small tight clusters and the cluster centroids.

Each partial cluster is represented by a single representative element. No new representatives will be created; only the current ones will be moved. CURE now clusters the partial clusters by iteratively merging the closest clusters. When two clusters are merged, their representative elements are shrunk towards the mean of the new cluster by a user specified shrinking factor α. The distance between two clusters is the minimum distance between a pair of representative points (one from each cluster). This is similar to single-link clustering except that only representative points are used. This similarity measure allows CURE to discover clusters of arbitrary shapes and sizes. Using only representative points reduces the effect of outliers since these points are continuously shrunk toward the center of the clusters. Figure 2.11 (c) shows the resulting clusters after this step. Notice that the crosses have been pulled toward the center of the clusters. At the final stage, each element from *X* that was not in the random sample is assigned to the cluster containing the representative element closest to it.

The time complexity of CURE is $O(n)$, making it efficient for large data sets. However, the algorithm is very sensitive to its input parameters: the shrinking factor α and the random sample size.

2.3.4 Rock

ROCK, RObust Clustering using linKs (Guha, Rastogi and Kyuseok 1999), is an algorithm for clustering binary and categorical data. Previous clustering methods that use a distance measure, such as the Euclidean distance between elements, are not suitable for binary and nominal data. For example, in CURE, it is hard to define the mean of a cluster of nominal feature vectors. Furthermore, consider a data set containing customers' transactions at a grocery store where each transaction is represented by a set of asymmetric binary variables describing each item available in the store. The feature space is large but each transaction may only have a small number of features instantiated to 1. Consider the cluster of purchased French delicacies such as {Bordeaux wine, Blue cheese, croissants, baguettes, etc.} It is common that two transactions in this cluster will have few common items and thus their distance will be large. However, there may exist another transaction in the cluster that overlaps with several features of both of them.

Most previous methods tend to prefer clusters of similar shapes. ROCK allows different shape clusters. Suppose d_{ij} gives the similarity between two elements x_i and x_j. A pair of elements x_i and x_j are neighbours if $d_{ij} \geq \theta$ for some fixed threshold θ. A typically used similarity function is the matching coefficient described in Section 1.2.2:

$$d_{ij} = \frac{A}{A+B} \qquad \text{(Eq. 2.4)}$$

where A is the number of matching features between x_i and x_j while B is the number of non-overlapping features between x_i and x_j. If a feature is missing from either x_i or x_j then that feature is considered non-overlapping.

The number of *links* between two elements x_i and x_j, l_{ij}, is defined as the number of their common neighbours. Links incorporate global information of neighbouring elements in the relationship between pairs of elements. This addresses the issue of the two transactions in our French delicacy example that share few items but that both share items with another transaction.

Let the aggregate interconnectivity of a cluster c, $I(c)$, be:

$$I(c)=\sum_{x_i,x_j\in c} l_{ij} \qquad \text{(Eq. 2.5)}$$

Intuitively, a good cluster *c* will have high aggregate interconnectivity. This ensures that elements with many common neighbours will be placed in the same cluster. However, this measure is maximized when all elements are placed in the same cluster. ROCK normalizes the aggregate interconnectivity by a static model of expected interconnectivity of a cluster:

$$I(c)=\sum_{x_i,x_j\in c} \frac{l_{ij}}{|c|^{1+2f(\theta)}} \qquad \text{(Eq. 2.6)}$$

where $|c|$ is the number of elements in cluster *c*, $|c|^{f(\theta)}$ is the expected number of neighbours in *c* and $|c|^{1+2f(\theta)}$ is then the expected number of links between elements in *c*.

ROCK first selects a random sample of elements and performs agglomerative hierarchical clustering (AGNES) using the following similarity computation (based solely on links):

$$d(c_i,c_j)=\frac{clinks(c_i,c_j)}{(|c_i|+|c_j|)^{1+2f(\theta)}-|c_i|^{1+2f(\theta)}-|c_j|^{1+2f(\theta)}} \qquad \text{(Eq. 2.7)}$$

where $clinks(c_i, c_j)$ is the number of cross-links between pairs of elements from clusters c_i and c_j. All elements that were not part of the random sample are then assigned to their closest cluster.

ROCK has worst-case time complexity of $O(n^2 + nm_m m_a + n^2 logn)$ where m_m is the maximum number of neighbours and m_a is the average number of neighbours. ROCK is a good algorithm for categorical data but its complexity makes it inefficient for large data sets.

2.3.5 Chameleon

CURE ignores the aggregate interconnectivity between two clusters while ROCK ignores the average closeness between clusters. Chameleon (Karypis, Han and Kumar 1999) combines the advantages of CURE and ROCK while employing dynamic modeling of clusters to improve clustering quality. Clusters are merged in Chameleon if they have high interconnectivity and closeness relative to each cluster's internal interconnectivity and closeness.

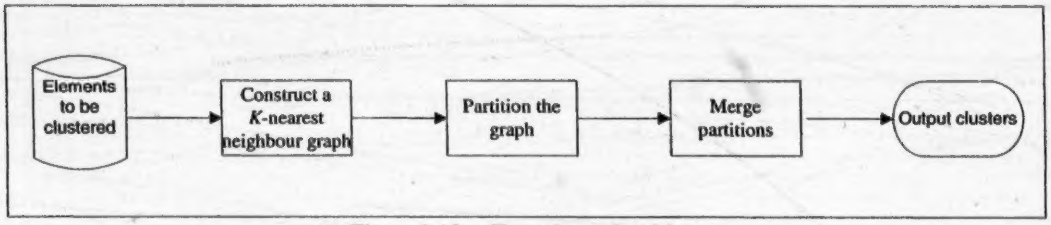


Figure 2.12 – Chameleon algorithm.

The steps of the Chameleon algorithm are illustrated in Figure 2.12. In the first step, a *K*-nearest neighbour graph is constructed. Each element to be clustered is represented by a vertex in the graph and a weighted edge between vertices represents the similarity between the two vertices. An edge is only present between two vertices x_i and x_j if x_j is one of the *k* most similar elements of x_i and vice versa. This data abstraction dramatically reduces the dimensionality of the data, making the algorithm more efficient.

The **absolute interconnectivity** between two clusters (subgraphs) G_1 and G_2, $AI(G_1, G_2)$, is defined as the aggregate similarity between the two clusters:

$$AI(G_1, G_2) = \sum_{x \in G_1} \sum_{y \in G_2} sim(x, y) \qquad \text{(Eq. 2.8)}$$

where $sim(x, y)$ is any similarity measure between two elements.

The **absolute closeness** between two clusters (subgraphs) G_1 and G_2, $AC(G_1, G_2)$, is defined as the average similarity between a pair of elements, one from each cluster:

$$AC(G_1, G_2) = \frac{1}{|G_1||G_2|} AI(G_1, G_2) \qquad \text{(Eq. 2.9)}$$

A difference between the absolute interconnectivity and the absolute closeness is that the latter takes zero similarity pairs into account. In Figure 2.13, the interconnectivity in (a) and (b) remains constant. However, the closeness in (a) is higher than in (b) since there are more zero similarity pairs in (b).

Let *mincut*(G) be the minimal edge bisection of a graph G. An even-sized partition $\{G', G''\}$ of a graph G is called a minimal edge bisection of G if $AI(G', G'')$ is minimal among all such partitions.

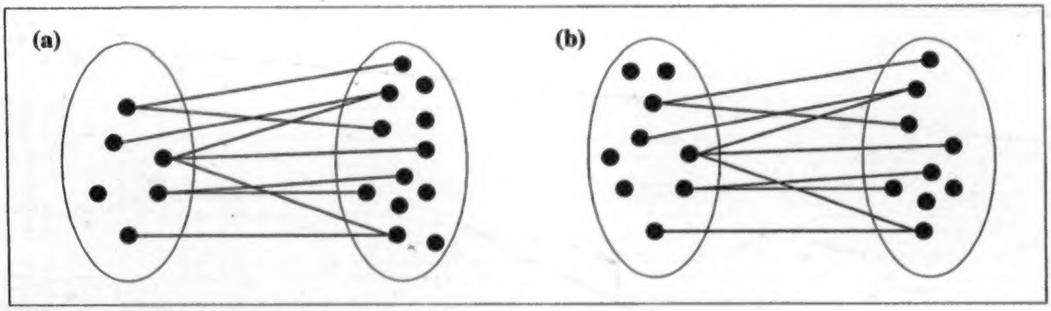


Figure 2.13 – Interconnectivity vs. closeness.
The interconnectivity between (a) and (b) is the same but the closeness changes.

Let *GL* be a list of graphs initially containing the *K*-nearest graph. In the second step of Chameleon, the *K*-nearest neighbour graph is iteratively partitioned by splitting the largest graph *G* from *GL* into two subgraphs using *mincut*(*G*) and replacing *G* by the two subgraphs in *GL*. This step repeats until *G* contains fewer than θ elements. At this point, the original *K*-nearest graph is partitioned into several small tight subgraphs.

In the final step of the algorithm, Chameleon iteratively merges the two subgraphs G_1 and G_2 from *GL* that maximize $gsim(G_1, G_2)$, where *gsim* is defined below in Eq. 2.12. The merged cluster is then placed in *GL* and this step repeats itself until $gsim(G_1, G_2) < \sigma$.

The similarity between clusters (subgraphs) G_1 and G_2, $gsim(G_1, G_2)$, combines the relative interconnectivity, $RI(G_1, G_2)$, and the relative closeness, $RC(G_1, G_2)$, between G_1 and G_2. The relative interconnectivity is obtained by normalizing the absolute interconnectivity by the internal interconnectivity of the individual clusters:

$$RI(G_1,G_2)=\frac{2AI(G_1,G_2)}{II(G_1)+II(G_2)} \qquad \text{(Eq. 2.10)}$$

where $II(G) = AI(G', G'')$ for $\{G', G''\}$ being the minimal edge bisection of *G*. This normalization is similar to the one in ROCK except that here we have a dynamic model of the expected interconnectivity between clusters. Similarly, we have:

$$RC(G_1,G_2)=\frac{AC(G_1,G_2)}{\frac{|G_1|}{|G_1|+|G_2|}IC(G_1)+\frac{|G_2|}{|G_1|+|G_2|}IC(G_2)} \qquad \text{(Eq. 2.11)}$$

where $IC(G) = AC(G', G'')$ for again $\{G', G''\}$ being the minimal edge bisection of G. The final similarity formula is:

$$gsim(G_1, G_2) = RI(G_1, G_2) \times RC(G_1, G_2)^{\alpha} \qquad \text{(Eq. 2.12)}$$

where α is a predefined combination parameter.

Chameleon has been shown to produce higher quality clusters than CURE but it suffers from a worst case time complexity of $O(n^2)$.

2.4 Other algorithms

There are several other well known families of algorithms. Density-based methods such as DBSCAN (Ester, Kriegel, Sander, Xu 1996) and OPTICS (Ankerst, Breunig, Kriegel, Sander 1999) discover clusters of dense elements that are separated by low density regions. Grid-based multi-resolution algorithms typically collect statistical information in grid cells and perform all clustering operations on these grids. CLIQUE (Agrawal, Gehrke, Gunopulos, Raghavan 1998) is a hybrid of grid and density methods. It is capable of handling high-dimensional data because all clustering operations are performed on the quantized space of the grid. Finally, model-based algorithms assume that the observed data points are generated by a mixture of underlying probability distributions. Usually, a mixture of Gaussians is assumed and the parameters of the individual Gaussians must be learned. Manning and Schütze (1999) provide a nice description of using the EM algorithm to learn the parameters.

Chapter 3

Resources

In this chapter, we describe the resources that are required for the CBC clustering algorithm. First, we describe WordNet, an online lexical hierarchy of concepts, and Minipar, a parser that will be used in our application of CBC to concept discovery. We will explain in detail the dependency trees output by Minipar as well as its lexicon. We proceed by presenting a data structure, called a feature database, which stores the features of elements to be clustered. Finally, we describe CBC's vector space model, a representation for our features, and the similarity model used for computing the similarity between two elements in CBC.

3.1 WordNet

WordNet[2] (Miller 1990) is an electronic dictionary organized as an acyclic graph. Each node in the graph, called a **synset**, represents a concept with an associated set of synonymous words. The arcs between synsets represent **hyponym/hypernym** (subclass/superclass) relationships[3] between concepts. In this dissertation, we use the term WordNet to refer to WordNet version 1.5.

---

[2] WordNet is available for download at http://www.cogsci.princeton.edu/~wn/.

[3] WordNet also contains other semantic relationships such as meronyms (part-whole relationships) and antonyms.

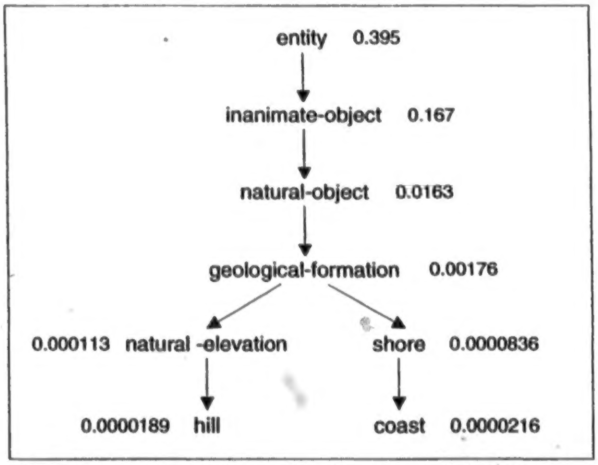


Figure 3.1 – Example hierarchy of synsets in WordNet.
The numbers beside the synsets represent their probability.

Figure 3.1 shows a fragment of WordNet. The number attached to a synset *s* is the probability that a randomly selected noun refers to an instance of *s* or any synset below it. These probabilities are not included in the original WordNet distribution. We use the frequency counts of synsets in the SemCor corpus (Landes, Leacock and Tengi 1998) to estimate them. Each word in SemCor is manually tagged with the WordNet sense that corresponds to the sense of the word. Since SemCor is a fairly small corpus (200K words), the frequency counts of the synsets in the lower part of the WordNet hierarchy are very sparse. We smooth the probabilities by assuming that all siblings are equally likely given the parent.

Lin (1997) defined a similarity measure between two WordNet synsets s_1 and s_2 as:

$$sim(s_1, s_2) = \frac{2 \times \log P(s)}{\log P(s_1) + \log P(s_2)} \qquad \text{(Eq. 3.1)}$$

where *s* is the most specific synset that subsumes s_1 and s_2. For example, using Figure 3.1, if s_1 = *hill* and s_2 = *shore* then *s* = *geological-formation* and *sim*(*hill*, *shore*) = 0.626.

WordNet is a general-purpose lexicon containing approximately 120,000 words organized into some 100,000 synsets. It includes many multi-word terms such as *act of God* and *Addison's syndrome* but lacks coverage of proper names. Although some are included, no serious attempt has been made to incorporate them. The coverage of words is similar to that of most dictionaries. However, it is the structure of the hierarchy of synsets that differentiates WordNet. One of its limitations is that it misses many domain specific senses of words. For example, WordNet does not include the user-interface-object sense of the word *dialog* (as often used in software manuals).

3.2 Minipar

Minipar[4] is a principle-based English parser (Berwick 1991). Like Principar (Lin 1993), Minipar represents its grammar as a network where nodes represent grammatical categories and links represent types of syntactic (dependency) relationships. The grammar network consists of 35 nodes and 59 links. Additional nodes and links are created dynamically to represent subcategories of verbs. Minipar employs a message passing algorithm that essentially implements distributed chart parsing. Instead of maintaining a single chart, each node in the grammar network maintains a chart containing partially built structures belonging to the grammatical category represented by the node. The grammatical principles are implemented as constraints associated with the nodes and links.

3.2.1 Parser internals

There are two major types of parsers: those that break the sentences into constituents and those that link individual words (dependencies). Minipar works with a constituency grammar internally, however the output of Minipar is a dependency tree. The idea of a constituent stems from the fact that a sentence is not just an ordered sequence of words, but that those words form groups (constituents) within the sentence. Example constituents include noun phrases, verb phrases, and prepositional phrases. A noun phrase can take many forms. It may consist of only a noun as in "*Carol* went to school", or a pronoun as in "*She* enunciates well", or a determiner followed by a noun as in "*The sun* is shining", etc. Constituents may also consist of other constituents. For example, a noun phrase may be a determiner followed by a noun and a prepositional phrase such as in "Henry ate *the salad with croutons*".

A constituency grammar is a context-free grammar with some optionally added information. Starting with a sentence symbol *S*, the grammar defines all legal sentences in the language. Figure 3.2 shows a small constituency grammar. A grammar consists of:

---

[4] Available at http://www.cs.ualberta.ca/~lindek/minipar.htm.

```
1. S  ← NP VP            S  ← NP:subj VP:h
2. NP ← N                NP ← N:h
3. NP ← NP PP            NP ← NP:h PP:mod
4. NP ← DET N            NP ← DET:det N:h
5. VP ← V NP             VP ← V:h NP:obj
6. PP ← P NP             PP ← P:h NP:pcomp
```

Figure 3.2 – Sample constituency grammar.
The left column shows the rewriting rules used to define all the legal sentences in the language; the right column shows the same rules augmented with relations (shown in italic).

- a set of terminal symbols, which are the possible parts-of-speech (e.g. noun, verb, adjective);
- a set of non-terminal symbols, which are the constituents (e.g. noun phrase, verb phrase);
- a starting symbol, which is the sentence symbol *S*; and
- a set of rewriting (or production) rules, which are rules for replacing a non-terminal symbol with a sequence of non-terminal and terminal symbols.

The terminal symbols in Figure 3.2 are *N*, *V*, and *P*, representing nouns, verbs, and prepositions respectively. The non-terminal symbols are *NP*, *VP*, and *PP* representing noun phrase, verb phrase, and prepositional phrase respectively. The six lines represent the rewriting rules. Rule 1 is the only one involving the sentence symbol *S* so the grammar covers all sentences that consist of a noun phrase followed by a verb phrase. It covers three ways in which noun phrases may be decomposed (rules 2-4). Below are sentences that are covered by this simple grammar; sentences that are not covered are marked with an asterix (*):

```
Gisele likes golf.
Grandma baked the pies with love.
*Hubert disliked the haircut that Leo gave him.
*Annette shopped with Denise.
The man saw a dog in the park with a telescope.
```

The third sentence is not covered by the grammar since the grammar does not allow relative clauses. In the fourth sentence, the prepositional phrase *with Denise* cannot be attached to the verb without adding a rule in the grammar such as *VP ← V PP*. The last example shows that even some complicated sentences are covered by the simple grammar of Figure 3.2. In that sentence, there are two different possible analyses. This is due to the ambiguity of the prepositional phrase

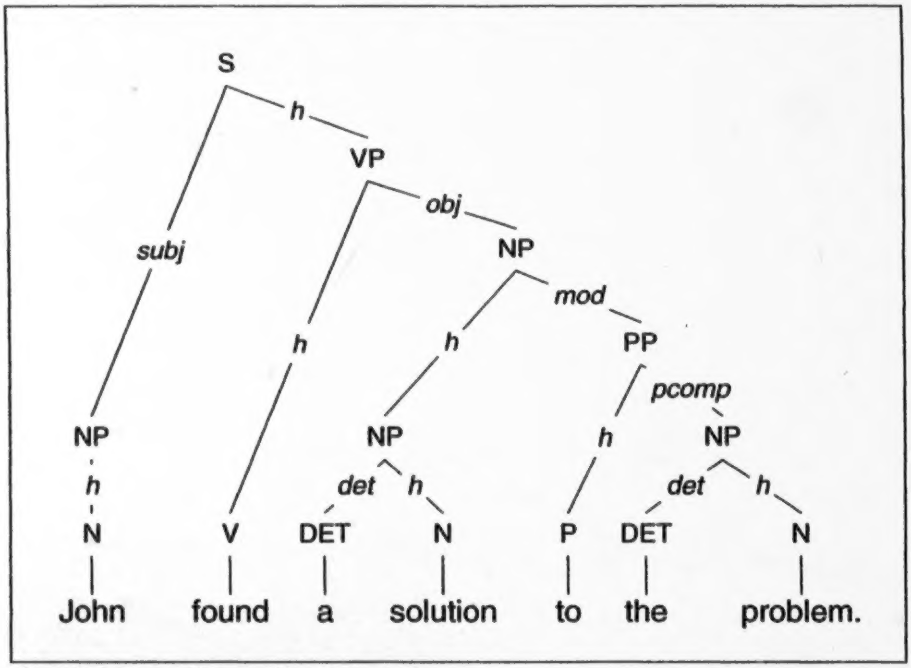


Figure 3.3 – Example constituency tree.
Constituency tree, using the grammar of Figure 3.2, for the sentence *John found a solution to the problem.*

attachment (Pantel and Lin 2000). The prepositional phrase *with a telescope* can either attach to the noun phrase *in the park* (meaning that the park is "that special park that has a telescope") or to the noun phrase *a dog in the park* (meaning that the dog has a telescope). However, the more semantically plausible attachment of *with a telescope* is to the verb phrase *the man saw* (meaning that the man is seeing a dog by looking through a telescope), which is not legal in this grammar.

A constituency grammar also allows the parser to enforce constraints between constituents. For example, we may enforce the agreement between the subject and the verb of a sentence such that if the subject is second person then the verb must be conjugated in the second person. In French, we may enforce the agreement in gender between a noun phrase and its adjective.

In Minipar, the constituency grammar was augmented with relations. An example is shown in the right column of Figure 3.2. Because these relations allow Minipar to identify the head of each constituent (e.g. the head of a noun phrase is the noun and the head of a prepositional phrase is the preposition), represented by the relation *h*, the conversion from a constituency tree to a dependency tree is straightforward. For each constituent (non-terminal) *C* in a constituency tree, we create $|C| - 1$ dependency relationships, where $|C|$ is the number of non-terminals and

Table 3.1 – A subset of the dependency relations in Minipar outputs.

RELATION	*DESCRIPTION*	*EXAMPLE*
appo	appositive of a noun	the CEO, **John**
det	determiner of a noun	**the** dog
gen	genitive modifier of a noun	**John's** dog
mod	adjunct modifier of any type of head	**tiny** hole
~~nn~~	prenominal modifier of a noun	**station** manager
pcomp	complement of a preposition	in the **garden**
subj	subject of a verb	**John** loves Mary.
sc	small clause complement of a verb	She forced him to **resign**

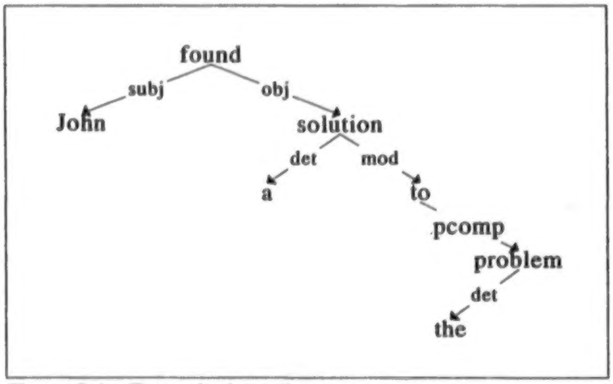
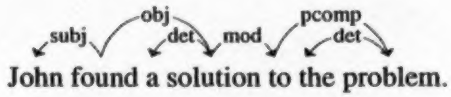


Figure 3.4 – Example dependency tree.
The dependency tree extracted by Minipar for the sentence *John found a solution to the problem.*

terminals in its rewrite rule. Each dependency relationship will be between the head of *C* and the head of one of the non-terminals in *C*'s rewrite rules, or between the head of *C* and the lexical entry linked to one of the terminals in *C*'s rewrite rules. The name of a dependency relationship will be the relation of the non-head link in *C*'s rewrite rule. Consider the constituency tree shown Figure 3.3. The dependency relationship *to* → *problem* with relation *pcomp* will be created for the *PP* non-terminal and *found* → *solution* with relation *obj* will be created for the *VP* non-terminal.

Figure 3.4 shows the resulting dependency tree after converting the constituency tree in Figure 3.3. The links in the diagram represent dependency relationships. The direction of a link is from the head to the modifier in the relationship. Labels associated with the links represent the relations that were added in Minipar's constituency grammar. Table 3.1 lists a subset of the dependency relations in Minipar outputs. The dependency tree in Figure 3.4 can also be represented in a more compact form as follows:



3.2.2 Lexicon

The lexicon in Minipar is derived from syntactic features (parts of speech and subcategorization frames) in WordNet. A subcategorization frame defines a particular syntactic category of a verb by listing the arguments (required modifiers) that the verb takes. They are essential in parsing to help attach the right arguments to the verbs. With additional proper names, the lexicon contains about 130,000 entries (in base forms). The lexicon entry of a word (or term) consists of the properties of its syntactic usages, the log of the ratio between the total number of words in the corpus and the frequency of the word, and the phrases in which the word is the head. The log ratios are obtained by parsing a corpus of text and accumulating the usages of words in the resulting parse trees. The lexical ambiguities are handled by the parser instead of a tagger.

Figure 3.5 shows a sample of some lexical entries. Each occurrence of a *syn* tag indicates a separate syntactic usage for a word. For example, in Figure 3.5, the word *algebra* has one syntactic usage, noun (*N*). That means that whenever the parser sees the word *algebra*, it will know that it is a noun. The *freq* tag indicates the log of the ratio between the total number of words in the corpus and the frequency of the noun usage of *algebra*. The *phrases* tag is used to enumerate the phrases in which *algebra* is the head: *Boolean algebra*, *linear algebra*, *matrix algebra*, etc. In the lexicon, a phrase such as *Boolean algebra* is written as "*algebra, Boolean*" so that the head of the phrase is always in front.

The word *cluster* in Figure 3.5 contains two usages. It may either be a noun (*N*) or a verb that may or may not take an object. Its noun usage log ratio was 10 whereas its verb usage was 14. The *syn* tag may contain more detailed information. For example, *Alberta*'s syntactic tag means that it is a proper noun (*PN*) and it has a semantic value of province (*+province*). *New York*'s single usage is as a compound noun (*+cn*) and proper name (*PN*) with semantic value city (*+city*). There are over 100 possible features in a *syn* tag, like *cn*, *PN*, etc.

```
(Alberta
  (syn (PN (sem (+province))))
  (freq N 9)
)
(algebra
  (syn (N))
  (freq N 11)
  (phrases (algebra, Boolean) (algebra, linear)
    (algebra, matrix) ...)
)
(cluster
  (syn (N))
  (syn (T[n]))
  (freq N 10 V_N_N 14)
)
(York, New
  (syn (+cn PN (sem (+city))))
)
(walk
  (syn (N))
  (syn (T[n]))
  (freq N 13 V_N_N 7)
  (phrases (walk of life) (walk out) (walk out of)
    (walk out on) ...)
)
```

Figure 3.5 – Sample entries from Minipar's lexicon.
Lexicon entries for the words *Alberta*, *algebra*, *cluster*, *New York* and *walk*.

3.2.3 Probability model

Like chart parsers, Minipar constructs all possible parses of an input sentence. However, only the highest ranking parse tree is output. Although the grammar is manually constructed, the selection of the best parse tree is guided by the statistical information obtained by parsing a 1GB newspaper corpus with Minipar. The statistical ranking of parse trees is based on the following probabilistic model. The probability of a dependency tree is defined as the product of the probabilities of the dependency relationships in the tree. Formally, given a tree *T* with root *root* consisting of *D* dependency relationships ($head_i$, $relationship_i$, $modifier_i$), the probability of *T* is given by:

$$P(T) = P(root)\prod_{i=1}^{D} P(relationship_i, modifier_i \mid head_i) \qquad \text{(Eq. 3.2)}$$

where $P(relationship_i, modifier_i \mid head_i)$ is obtained using Maximum Likelihood Estimation (MLE).

Minipar parses newspaper text at about 500 words per second on a Pentium-III 750Mhz with 500MB memory. Evaluation with the manually parsed SUSANNE corpus (Sampson 1995) shows that about 89% of the dependency relationships in Minipar outputs are correct (Lin 1998c). The recall of Minipar outputs, defined as the percentage of dependency relationships in the SUSANNE corpus that are extracted by Minipar, varies a great deal depending on the genre of the input document, from 80% (novels) to 87% (news reportage). This accuracy is comparable to other state-of-the-art broad coverage English parsers (Collins 1996; Charniak 2000).

3.3 Similarity measure

In this section, we describe the vector space model used by CBC and the data structure, called a feature database, used to retrieve the features of an element. The use of a database is motivated by the need to deal with a large number of elements and features. The features that will be used in the document clustering and concept discovery applications will also be explained. We then proceed by describing the similarity model used in CBC for computing the similarity between two elements.

3.3.1 Feature database

The elements to be clustered are represented by a feature vector, which is a series of measurements that quantitively describe the elements. For example, you might describe a set of professional basketball players by their height, age, and average points, rebounds, assists, and blocks per game. If you wish to describe the buying patterns of a set of customers, you might measure the number of individual products purchased in each transaction. The feature vectors are used to compute the similarity between elements. Given an element, the feature database is used to retrieve the features of the element along with their measurements. We now describe the data structure used for storing the feature database.

```
Haaretz 180
    -V:in:N
         write          3        6.033
         observe        1        4.857
         quote          1        4.141
    -N:in:N
         report         3        6.966
         headline       1        4.787
    -N:for:N
         columnist      1        4.561
         source         1        4.229
         correspondent  1        4.163
    -N:of:N
         editor         1        3.565
         issue          1        3.290
    -V:subj:N
         quote          16       7.496
         report         10       5.595
         say            40       3.956
         experiment     1        3.631
         publish        1        2.759
         launch         1        2.588
         predict        1        2.506
         complain       1        2.376
         suggest        1        2.115
    -V:obj:N
         tell           5        4.722
         quote          1        3.102
         catch          1        3.024
         beat           1        2.880
         report         1        2.699
         say            1        2.212
    -N:nn:N
         newspaper      20       8.801
         Daily          8        8.061
         report         3        4.814
         interview      1        3.466
    N:nn:N
         newspaper      22       9.090
         Hebrew         2        6.871
```

Figure 3.6 – Feature database entry excerpt for the word *Haaretz*. The feature database was built using Minipar and a 1GB newspaper corpus.

Triples

Although the implementation of the feature database is meant to be general enough to represent any data set, it was designed with the applications of document clustering and concept discovery in mind. As long as the features for a clustering problem can be stored in the feature database, CBC is directly applicable.

The features that we will use for clustering documents are the words that occur within each document. Slightly more complicated, the features for concept discovery will be the grammatical contexts in which each word occurs. In the parsed sentence of Figure 3.4, we can extract six features; one for each dependency relationship (or context). One such context is that *John* is the subject of the verb *found*. We write this in the form of a triple as (*John*, *subj-of*, *found*), where *John* is the element and the feature is the relationship *subj-of* and the word *found*. Entries in the feature database will contain these triples along with their frequency. Suppose for document clustering we want to add the feature *word* for the document *doc*. We will then create a dummy relationship called *contains* and add the triple (*doc*, *contains*, *word*) to the feature database.

Figure 3.6 shows an example of the format that will be used throughout the dissertation to describe an entry in the feature database. It shows an excerpt of the entry for the word *Haaretz* using a 1GB newspaper corpus. The frequency counts, in the second column, are obtained by counting the triples in the parser output of Minipar on the corpus. The numbers in the third column are pointwise mutual-information scores. Mutual-information is described in Section 3.3.2. The number beside *Haaretz* is the total frequency of the word in the corpus. Try looking at the features in Figure 3.6 to determine what Haaretz is. It might be obvious to you that *Haaretz* is a newspaper[5].

Implementation

The implementation of the feature database consists of two hash tables. One is a persistent hash table on disk and the other is a hash table in memory. A hash table, also called a direct access table, is an efficient data structure for inserting and searching elements under certain conditions. The hash table on disk contains the full database of element-feature mappings. The hash table in memory is used as a cache and is initially empty. When a request for an element is made, first the cache is verified. If the element exists, then it is returned without going to disk. If it does not exist, then the hash table on disk is verified. If the element is not found, then NULL is returned. Otherwise, the element is returned and the cache is updated with the element. Each cache entry also contains a Boolean flag indicating whether the entry was changed. When an element is purged from the cache, the disk entry will be updated only if this flag indicates that the entry has changed.

---

[5] *Haaretz* is an Israeli newspaper that was founded in Jerusalem in 1919 by a group of Zionist immigrants, mainly from Russia.

A hash function, $h(e)$, is a function that maps an element e to a bucket in the hash table. A perfect hash function is one where each element will be mapped to a unique bucket. Unfortunately, a perfect hash function rarely exists. In practice, a good hash function maps most elements to unique buckets in the hash table, but occasionally maps a small number of elements to the same bucket. When this occurs, it is called a collision. If not many collisions occur, then a hash table is searchable in effectively constant time. If too many collisions occur, then another data structure such as a binary tree should be used since the hash table search time may become linear.

To handle collisions, we store a linked list at each bucket in the hash table. When elements are inserted into the hash table, they are added to one of the linked lists. Assuming that elements are unique, insertion is made in constant time since we only need to do a table lookup for a bucket and add the element to the front or back of the linked list. Searching for an element, however, requires a linear search of a linked list. If a hash function mapped all elements to the same bucket, searching would be $O(n)$, where n is the number of elements. Other collision-handling algorithms apply a second hash function after a collision (re-hashing) or look at the next sequential bucket in the hash table after a collision (linear probing).

In document clustering and concept discovery, the elements are strings. The hash function $h(e)$ must therefore map strings to buckets in the hash table. A good strategy for designing hash functions for strings is to first determine a positive integer h from the characters of the string and then take h modulo m, where m is the size of the hash table. A simple function for h simply adds up the integer value of each character in the string. We use the hash function *hashpjw* (Aho et al. 1986, pp. 435-436):

```
h = 0;
for each character c in the string {
   shift the bits of h four positions to the left;
   h = h + c;
   if any of the four high-order bits of h equals 1 {
      shift the four high-order bits of h 24 positions to the
      right
      exclusive-or the four shifted bits into h
      reset to 0 the four high-order bits of h
   }
}
```

hashpjw was shown to distribute strings consistently well in many different table sizes.

The feature database also has inverse indexing allowing efficient retrieval of all elements that contain a given feature. Suppose we wish to add a feature for the word *emblazon* in the sentence:

`Guzman's umbrella was emblazoned with cartoon character Bart Simpson -- a character that clearly remains popular in the school supplies market.`

Minipar will analyze *emblazon* as the head of an object relation with *umbrella* (shorthand is *V*:*obj*:*N*). Therefore the triple (*emblazon*, *V*:*obj*:*N*, *umbrella*) will be added to the feature database in the entry for *emblazon*. We will also add the feature "-*V*:*obj*:*N emblazon*" in the database for the *umbrella* entry; the triple is (*umbrella*, -*V*:*obj*:*N*, *emblazon*). The (-) sign indicates the inverse relationship. In the context of grammatical relationships, the inverse implies that the head and modifier of the relationship are inverted. More examples of inverse relationships are shown in Figure 3.6. The inverse relationship allows us to ask a question like "what else can you do to umbrellas?" Figure 3.7 shows the most frequent verbs that occurred in a 1GB newspaper corpus with object *umbrella*.

3.3.2 Vector-space model

The feature database described in the previous section is used to store and retrieve the features of elements. In this section, we describe the feature vectors used in CBC, called the vector-space model. It will be used to describe documents in our document clustering system and words in our concept discovery system.

In document clustering, each document can be represented by the terms that occur within it and the value of a feature is a statistic of the term. The statistic can simply be 1 if the term occurs in the document; or 0 otherwise. This is called the Set-of-Words model. Or, the statistic can be

```
umbrella  892
    -V:obj:N            251       0.715
         carry          14        5.118
         open           11        4.964
         use            11        3.590
         provide        7         3.723
         hold           7         3.512
         like           6         3.317
         unfurl         5         7.765
         fall under     5         7.316
         install        5         4.948
         bring          5         3.364
         put            5         3.051
         have           5         1.353
         emblazon       4         6.924
         place          4         3.580
         sell           4         3.008
         erect          3         5.087
         plant          3         4.384
         test           3         3.856
         extend         3         3.791
         choose         3         3.573
         create         3         2.865
         send           3         2.698
         furl           2         6.778
         wield          2         4.618
         swallow        2         4.535
         wave           2         4.057
         manufacture    2         3.817
         retain         2         3.402
         serve as       2         3.027
         look at        2         2.753
         design         2         2.696
         close          2         2.566
         schedule       2         2.447
```

Figure 3.7 – Feature database entry excerpt for the word *umbrella*. Shows the verbs that take *umbrella* as object. The feature database was built using Minipar and a 1GB newspaper corpus.

the term's frequency, *tf*, within the document. This is called the Bag-of-Words model. Figure 3.8 shows a subset of the features in Chapter 1 of this dissertation using the Bag-of-Words model. The numbers in the right column represent frequency counts.

The feature database can be used to retrieve the frequency of a particular word in a document. In Figure 3.8, the features in bold are not very informative as to the subject of the document. In order to discount such terms with low discriminating power, *tf* is usually combined with the term's inverse document frequency, *idf*, which is the inverse of the percentage of documents in which the term occurs. The assumption is that informative words will occur many times within a

```
Chapter 1          3694
     The           185
     of            142
     a             114
     is            79
     and           75
     to            73
     Clustering    66
     features      53
     in            47
     feature       46
     are           44
     elements      39
     be            36
     For           34
     as            25
     that          24
     by            21
     data          20
     we            19
     an            18
     ...
```

Figure 3.8 – Document clustering features using the Bag-of-Words model.
Subset of the features in Chapter 1 of this dissertation.

document but not in many other documents. This measure is referred to as *tf-idf* (Salton and McGill 1983):

$$tf\text{-}idf = tf \times \log idf \qquad \text{(Eq. 3.3)}$$

Mutual Information

Mutual information is a commonly used measure for the association strength between two words (Church and Hanks 1989). The pointwise mutual information (Manning and Schütze 1999) between two events x and y, mi_{xy}, is given by:

$$mi_{xy} = \log \frac{P(x,y)}{P(x)P(y)} \qquad \text{(Eq. 3.4)}$$

Mutual information compares two models for predicting the co-occurrence of x and y: one is the MLE of the joint probability of x and y and the other is some baseline model. In the above equation, the baseline model assumes that x and y are independent. Mutual information is high when x and y occur together more often than by chance. Note that in information theory, mutual information refers to the mutual information between two random variables rather than between two events as used in this dissertation. The mutual information between two random variables X and Y, MI_{XY}, is given by:

$$MI_{XY} = \sum_{x \in X} \sum_{y \in Y} P(x, y) \log \frac{P(x, y)}{P(x)P(y)} \qquad \text{(Eq. 3.5)}$$

The mutual information between two random variables is the weighted average of the pointwise mutual information of all possible combinations of events involving the two variables. It measures the amount of information one variable gives about another.

Mutual Information vector-space model

In CBC, for each element e, we construct a **frequency count vector** $C(e) = (c_{e1}, c_{e2}, \ldots, c_{em})$, where m is the total number of features and c_{ef} is the frequency count of feature f occurring in element e. In document clustering, e is a document and c_{ef} is the frequency of term f in e. In concept discovery, e is a word and c_{ef} is the number of times e occurred in context f. Once we have collected these features in the database, we perform feature extraction by constructing a **mutual information vector** $MI(e) = (mi_{e1}, mi_{e2}, \ldots, mi_{em})$ for each element e, where mi_{ef} is the pointwise mutual information between element e and feature f, which is defined as:

$$mi_{ef} = \log \frac{\frac{c_{ef}}{N}}{\frac{\sum_{i=1}^{n} c_{if}}{N} \times \frac{\sum_{j=1}^{m} c_{ej}}{N}} \qquad \text{(Eq. 3.6)}$$

where n is the number of elements to be clustered and $N = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij}$ is the total frequency count of all features of all elements. The experiments presented in Section 6.1.3 illustrate that using the mutual-information vector $MI(e)$ produces much higher quality clusters then by using the term frequency vector $C(e)$.

A well-known problem with mutual information is that it is biased towards infrequent elements/features. We therefore multiplied mi_{ef} with a discounting factor:

$$\frac{c_{ef}}{c_{ef}+1} \times \frac{\min\left(\sum_{i=1}^{n} c_{ei}, \sum_{j=1}^{m} c_{jf}\right)}{\min\left(\sum_{i=1}^{n} c_{ei}, \sum_{j=1}^{m} c_{jf}\right)+1} \qquad \text{(Eq. 3.7)}$$

3.3.3 Similarity model

Any similarity metric may be applied in CBC as long as it can measure the similarity between two feature vectors. In Section 1.2.2, we presented some commonly used similarity measures. Other measurements that are applicable in CBC include the Dice coefficient (Frakes and Baeza-Yates 1992) and Lin's information theoretic similarity (Lin 1998b). In CBC, we will compute the similarity between two elements e_i and e_j using the *cosine coefficient* (Salton and McGill 1983) of their mutual information vectors:

$$sim(e_i, e_j) = \frac{\sum_f mi_{e_if} \times mi_{e_jf}}{\sqrt{\sum_f mi_{e_if}^2 \times \sum_f mi_{e_jf}^2}} \qquad \text{(Eq. 3.8)}$$

The cosine coefficient is a common similarity model because of its simplicity. However, it is only applicable with numerical features. The applications of CBC proposed in this dissertation all deal with numerical features and hence the cosine coefficient is applicable.

Chapter 4

CBC

In this chapter, we describe the Clustering by Committee (CBC) algorithm. CBC is designed to address the general goal of clustering, which is to group data elements such that the intra-group similarities are high and the inter-group similarities are low. We first provide a motivation for CBC and then present two versions of the algorithm: a hard clustering version where each element is assigned to only one cluster and a soft clustering version where elements may be assigned to multiple clusters. We proceed by comparing CBC with its predecessor, UNICON, and by describing two applications of CBC: document clustering and concept discovery.

4.1 Motivation

CBC was motivated by a desire to automatically extract concepts and word senses from large collections of text. Broad-coverage lexical resources such as WordNet, described in Section 3.1, are extremely useful in applications such as word sense disambiguation (Leacock, Chodorow and Miller 1998) and question answering (Pasca and Harabagiu 2001). In previous approaches, word senses are usually defined using a manually constructed lexicon such as WordNet. There are several disadvantages associated with such lexicons. First, manually created lexicons often contain rare senses. For example, WordNet included a rare sense of *computer* that means 'the person who computes'. Using WordNet to expand queries to an information retrieval system, the expansion of *computer* will include words like *estimator* and *reckoner*. Also, the words *dog, computer* and *company* all have a sense that is a hyponym of *person* (a hyponym is defined in Section 3.1). Such rare senses make it difficult for a coreference resolution system to use WordNet to enforce the constraint that personal pronouns (e.g. *he* or *she*) must refer to a person. The second problem with these lexicons is that they miss many domain specific senses. For example, WordNet misses the user-interface-object sense of the word *dialog* (as often used in software manuals). One way to deal with these problems is to use a clustering algorithm to automatically induce semantic classes (Lin and Pantel 2001a).

Many clustering algorithms represent a cluster by the centroid of all of its members (e.g., K-means) (McQueen 1967) or by a representative element (e.g., K-medoids) (Kaufmann and Rousseeuw 1987). When averaging over all elements in a cluster, the centroid of a cluster may be unduly influenced by elements that only marginally belong to the cluster or by elements that also belong to other clusters. For example, when clustering words, we can use the contexts of the words as features and group together the words that tend to appear in similar contexts. For instance, U.S. state names can be clustered this way because they tend to appear in the following contexts:

List A

___ appellate court
___ capital
___ driver's license
___ outlaws sth.
___'s sales tax
campaign in ___
governor of ___
illegal in ___
primary in ___
senator for ___

If we create a centroid of all the state names, the centroid will also contain features such as:

List B

___'s airport
___'s business district
___'s mayor
___'s subway
archbishop of ___
fly to ___
mayor of ___
outskirts of ___

because some of the state names (like *New York* and *Washington*) are also names of cities.

Using a single representative from a cluster may be problematic too because each individual element has its own idiosyncrasies that may not be shared by other members of the cluster.

In CBC, the centroid of a cluster is constructed by averaging the feature vectors of a subset of the cluster members. The subset is viewed as a committee that determines which other elements belong to the cluster. By carefully choosing committee members, the features of the centroid tend to be the more typical features of the target class. For example, CBC chose the following committee members to compute the centroid of its state cluster: Illinois, Michigan, Minnesota, Iowa, Wisconsin, Indiana, Nebraska and Vermont. As a result, the centroid contains only features like those illustrated in List A. Cities are less likely to be attracted to the centroid.

4.2 Algorithm

CBC consists of three phases. In Phase I, we compute each element's top-*k* similar elements, for some small value of *k*. In our experiments, we used *k* in the range of [10, 20]. In Phase II, we construct a collection of tight clusters using the top-*k* similar elements from Phase I, where the elements of each cluster form a **committee**. The algorithm tries to form as many committees as possible on the condition that each newly formed committee is not very similar to any existing committee. If the condition is violated, the committee is simply discarded. In the final phase of the algorithm, each element *e* is assigned to its most similar clusters.

The framework of CBC allows it to cluster any set of elements as long as their features may be placed in the feature database described in Section 3.3.1. It should be noted that choosing the right features for a given clustering problem is critical to the success of any clustering algorithm.

4.2.1 Phase I

Computing the complete similarity matrix between pairs of elements is obviously quadratic. However, one can dramatically reduce the running time by taking advantage of the fact that the feature vector is sparse. By indexing the features, one can retrieve the set of elements that have a given feature (see Section 3.3.1). To compute the top similar elements of an element *e*, we first sort the features according to their pointwise mutual information values and then only consider a subset of the features with highest mutual information. Finally, we compute the pairwise similarity between *e* and the elements that share a feature from this subset. Since high mutual information features tend not to occur in many elements, we only need to compute a fraction of the possible pairwise combinations. Using this heuristic, similar words that share only low mutual

account	
checking account	0.145
bank account	0.143
fund	0.130
savings account	0.124
report	0.115
story	0.105
deposit	0.097
description	0.096
statement	0.095
testimony	0.093
trust fund	0.092
loan	0.090
claim	0.089
investment	0.088
assessment	0.085
document	0.084
asset	0.084
transaction	0.084
certificate of deposit	0.084
detail	0.083
duty	
responsibility	0.143
obligation	0.119
job	0.114
chore	0.100
assignment	0.099
task	0.099
role	0.095
post	0.092
military service	0.091
position	0.090
function	0.089
service	0.084
Patrol	0.083
work	0.080
mission	0.078
tariff	0.075
action	0.071
stint	0.069
prerogative	0.069
activity	0.068

Figure 4.1 – Top-20 similar words of *account* and *duty*.
The similar words were obtained from the TREC corpus using Lin's similarity measure (Lin 1998b).

information features will be missed by our algorithm. However, in our experiments, this had no visible impact on cluster quality (see Figure 6.1).

The similarity matrix is stored in a hash table similar to the feature database presented in Section 3.3.1. By querying an element in the hash table (e.g. a document or a word), we retrieve in constant time its top-*k* similar elements along with their similarity scores. The similarity between two elements is simply the cosine of the elements' feature vectors (see Section 3.3.3).

Figure 4.1 shows a sample of a similarity matrix, constructed using the TREC corpus described in Section 6.2.1, for the words *account* and *duty*. Notice that the lists of similar words do not distinguish between the senses of a word. For *account*, we find similar words for its *bank account* sense and its *story* sense. For *duty*, there are similar words for its *responsibility* and *tax* senses.

4.2.2 Phase II

The second phase of the clustering algorithm recursively finds tight clusters scattered in the similarity space. In each recursive step, the algorithm finds a set of tight clusters, called committees, and identifies residue elements that are not covered by any committee. We say a committee **covers** an element if the element's similarity to the centroid of the committee exceeds some high similarity threshold. The algorithm then recursively attempts to find more committees among the residue elements. The output of the algorithm is the union of all committees found in each recursive step. The details of Phase II are presented in Figure 4.2.

The highest scoring clusters at the end of Step 1 represent candidate committees. The score reflects a preference for bigger and tighter clusters. After clustering the similar element of *e*, only the highest scoring cluster is kept as a candidate committee. However, it is possible that a lower scoring cluster will be accepted as a committee in the recursion of Phase II (Step 6). The candidates have high intra-cluster similarity but they do not have low inter-cluster similarity (i.e. some candidates are similar to each other).

Step 2 gives preference to the larger and tighter candidate committees for Step 3, where a candidate committee is only kept if its similarity to all previously kept candidates is below a fixed threshold. This assures that the inter-cluster similarities are low. In our experiments, we set $\theta_1 = 0.35$. Step 4 terminates the recursion if no committee is found in the previous step.

```
Input:   A list of elements $E$ to be clustered, a similarity database $S$ from
         Phase I, thresholds $\theta_1$ and $\theta_2$.
Step 1:  For each element $e \in E$
             Cluster the top similar elements of $e$ from $S$ using average-link
             clustering (see Section 2.1.1).
             For each discovered cluster $c$, compute the following score:
             $|c| \times \text{avgsim}(c)$, where $|c|$ is the number of elements in $c$ and
             $\text{avgsim}(c)$ is the average pairwise similarity between elements in $c$.
             Store the highest-scoring cluster in a list $L$.
Step 2:  Sort the clusters in $L$ in descending order of their scores.
Step 3:  Let $C$ be a list of committees, initially empty.
         For each cluster $c \in L$ in sorted order
             Compute the centroid of $c$ by averaging the feature vectors of its
             elements and computing the mutual information scores in the same
             way as we did for individual elements.
             If $c$'s similarity to the centroid of each committee previously added
             to $C$ is below a threshold $\theta_1$, add $c$ to $C$.
Step 4:  If $C$ is empty, we are done and return $C$.
Step 5:  For each element $e \in E$
             If $e$'s similarity to every committee in $C$ is below threshold $\theta_2$, add
             $e$ to a list of residues $R$.
Step 6:  If $R$ is empty, we are done and return $C$.
         Otherwise, return the union of $C$ and the output of a recursive call
         to Phase II using the same input except replacing $E$ with $R$.
Output:  a list of committees.
```

Figure 4.2 – Phase II of CBC.

The residue elements are identified in Step 5. An element is covered if its similarity to the centroid of any committee exceeds some high similarity threshold; we used $\theta_2 = 0.25$. Every other element is a residue element. If no residues are found, the algorithm terminates; otherwise, we recursively apply the algorithm to the residue elements. In the recursive step, the top-*k* similar elements of element *e* will no longer be the same since the similar elements must be part of the residue elements. Therefore, the highest scoring clusters of Step 1 may result in the discovery of new committees. For example, suppose that the highest scoring cluster of the similar words of *bank* is (*financial institution*, *corporation*, *business*, *organization*) but that it is rejected in Step 3 because a similar candidate committee was previously accepted. Now suppose that *bank* is part of the residue elements and that in the recursive step its most similar words are (*shore*, *coast*, *coastline*, *shoreline*). The recursive step gives CBC the opportunity to discover this concept (although it might have already been discovered it in the previous recursion of Phase II by a word other than *bank*, like *shore*).

Each committee that is discovered in Phase II of CBC defines one of the final output clusters of the algorithm. Below are some sample committees discovered by CBC using the TREC corpus:

- `basketball, soccer, volleyball, softball, tennis, hockey, football, water polo, baseball, golf, badminton`
- `pink, purple, mauve, yellow, turquoise, blue, red, lavender, green, beige`
- `tulip, peony, carnation, daffodil, iris, lilac, chrysanthemum, orchid`
- `apricot, pear, nectarine, peach, plum, mango, fig, cherry, persimmon, melon, prune`

As discussed in Section 4.1, the words in the committees tend to be words that are not polysemous[6]. For example, in the *color* and *fruit* committees above, *orange* is omitted because of its polysemous use in the corpus. If *orange* were included in the *color* committee, then that committee may later incorrectly attract fruits to it.

---

[6] The words may in fact be polysemous words but only one sense tends to be utilized in the domain of the corpus used to discover the committees.

4.2.3 Phase III

Phase III has two different versions: one for outputting a hard clustering and one for a soft clustering. We use the hard clustering version for document clustering and the soft clustering for discovering word senses.

Hard-clustering version

In the hard-clustering version, every element is simply assigned to the cluster containing the committee to which it is most similar. This version resembles *K*-means in that every element is assigned to its closest centroid. Unlike *K*-means, the number of clusters is not fixed and the centroids do not change (i.e. when an element is added to a cluster, it is not added to the committee of that cluster).

Soft-clustering version

In the soft-clustering version, each element *e* is assigned to its most similar clusters in the following way:

```
let C be a list of clusters initially empty
let S be the top-200 similar clusters to e
while S is not empty {
   let $c \in S$ be the most similar cluster to e
   if similarity(e, c) < $\sigma$
      exit the loop
   if c is not similar to any cluster in C {
      assign e to c
      remove from e its features that overlap with the features
of c;
      add c to C
   }
   remove c from S
}
```

When computing the similarity between a cluster and an element (or another cluster) we use the centroid of the committee members as the representation for the cluster. The key to the soft-clustering version of the algorithm for discovering word senses is that once an element *e* is assigned to a cluster *c*, the intersecting features between *e* and *c* are removed from *e*. This allows CBC to discover the less frequent senses of a word and to avoid discovering duplicate senses.

1. facility
2. factory
3. reactor
4. refinery
5. power plant
6. site
7. manufacturing plant
8. **tree**
9. building
10. complex
11. landfill
12. dump
13. project
14. mill
15. airport
16. station
17. farm
18. operation
19. warehouse
20. company
21. home
22. center
23. lab
24. store
25. industry
26. park
27. house
28. business
29. incinerator
30. power station

Figure 4.3 – Top-30 similar words of *plant*.
The similar words were obtained from the TREC corpus using Lin's similarity measure (Lin 1998b). The word *tree* in bold is the only word that represents the *life* sense of *plant*.

For example, consider the clusters to which the word *plant* would be assigned. The top-30 similar words of *plant* are shown in Figure 4.3. Suppose there exists the following two clusters:

C_1: ground cover, perennial, shrub, bulb, annual, wildflower, shrubbery, fern, grass, ...

C_2: factory, power plant, refinery, power station, reactor, oil refinery, facility, manufacturing plant, ...

It is clear by Figure 4.3 that *plant* is often used as a *factory* in the training corpus. Only one of the top-30 similar words of *plant* refers to its life sense (this is the bold word *tree* shown in Figure 4.3). The feature vector of *plant* is dominated by features like (*-V:obj:N*, *operate*), (*-N:nn:N*, *owner*), and (*-V:subj:N*, *produce*). The similarity between *plant* and C_1 will therefore be very low. Most clustering algorithms would be unable to assign the word *plant* to C_1.

CBC's assignment algorithm will first assign *plant* to its most similar cluster C_2. It then removes the features that intersect from *plant*'s feature vector and C_2's feature vector. Below are some of the features of C_2:

C_2: __ project, __ owner, build __, close __, __ produces, __ makes, __ operates, ...

Removing the intersecting features can be viewed as removing the *factory* sense from the word *plant*. After removal, the similarity between the revised feature vector of *plant* and C_1 greatly increases, allowing CBC to discover *plant*'s life sense.

4.3 Comparison with UNICON

UNICON (Lin and Pantel 2001a) also constructs cluster centroids using a small set of similar elements like the committees in CBC. One of the main differences between UNICON and CBC is that UNICON only guarantees that the committees do not have overlapping members. However, the centroids of two committees may still be quite similar. For example, the following two committees generated by UNICON do not share any members. However, their centroids are very similar.

- `Harvard University, Harvard, Stanford University, University of Chicago, Columbia University, New York University, University of Michigan, Yale university, MIT, University of Pennsylvania, Cornell University`
- `University of Rochester, University of Miami, University of Colorado, Ohio State University, University of Florida, Harvard Medical School, University of North Carolina, University of Houston`

UNICON deals with this problem by merging such clusters. In contrast, Step 2 in Phase II of CBC only outputs a committee if its centroid is not similar to any previously output committee.

Another main difference between UNICON and CBC is in Phase III of CBC. UNICON has difficulty discovering senses of a word when this word has a dominating sense. For example, UNICON cannot discover *plant*'s life sense as described in the previous section.

4.4 Applications

In our experiments, we will apply CBC to the tasks of clustering documents and discovering concepts. Below, we describe these applications. For each clustering application, the feature representation is very important. After collecting the features in the feature database described in Section 3.3.1, CBC may be directly applied.

4.4.1 Document clustering

The goal of document clustering is to discover documents with similar topics. It is useful in many information retrieval tasks. Document clustering was initially proposed for improving the precision and recall of information retrieval systems (van Rijsbergen 1979). Because clustering is often too slow for large corpora and had indifferent performance for this application (Jardine and van Rijsbergen 1971), document clustering has been used more recently in document browsing (Cutting, Karger, Pedersen and Tukey 1992), to improve the organization and viewing of retrieval results (Hearst and Pedersen 1996), to accelerate nearest-neighbor search (Buckley and Lewit 1985) and to generate Yahoo-like hierarchies (Koller and Sahami 1997). Common characteristics of document clustering include:

- there is a large number of documents to be clustered;
- the number of output clusters may be large;
- each document has a large number of features (e.g. the features may include all of the terms in the document); and
- the feature space, the union of the features of all documents, is even larger (e.g. all possible English words).

We use the Bag-of-Words model and represent a document by the stemmed words that occur within it. Stemming is the process of removing the common morphological and inflexional endings from words. It is used to reduce terms to a common-base form and consequently reduce the data sparseness. We use a standard stemming algorithm used in *IR* called Porter's stemmer (Porter 1980). For example, this stemmer will stem the word *answering* to *answer* and the word *feature* to *featur*. Given a corpus of documents, we use a tokenizer to collect the frequency counts of all stemmed words and we insert these counts along with pointwise mutual information scores into the feature database described in Section 3.3.1.

One of the datasets that we experimented with consists of 18,828 newsgroup articles partitioned nearly evenly across 20 different newsgroups. Each file contains one article and the pathname of the file represents the newsgroup of the article. For example, rec.sport.baseball/102663 is an article about the uniforms of certain teams in Major League Baseball while talk.politics.mideast/76234 discusses Israel's occupation of the West Bank, Gaza, and Golan. Here is a sample cluster discovered by CBC on this dataset:

```
sci.med/59498, sci.med/59474, sci.med/59641, sci.med/59167,
sci.med/59080, sci.med/59459, sci.med/59490, sci.med/59511,
sci.med/59191, sci.med/59447, sci.med/59432, sci.med/59244,
sci.med/59487, sci.med/59393, alt.atheism/54136
```

Here are the top-15 features (stemmed words) of this cluster according to mutual-information scores:

```
stone, kidnei, pain, calcium, oxal, doctor, prevent, intak,
help, wai, treatment, told, patient, rai, effect
```

By looking at these features, a good guess is that the documents in the cluster are about kidney stones. In fact, each sci.med/* document is about patients discussing kidney stones. The last document in this cluster, alt.atheism/54136, seems out of place but in fact does refer to kidney stones. Here is the content of that document:

```
In article <1993Apr26.000410.18114@daffy.cs.wisc.edu>
writes
> In article <C62B52.LKz@blaze.cs.jhu.edu>
arromdee@jyusenkyou.cs.jhu.edu (Ken Arromdee) writes:
> >I can think of a lot more agonizing ways to get killed.
> >Fatal cancer, for instance.
> >
> >Anyone else have some more?  Maybe we can make a list.
> How about dying of a blood clot in a _very_ bad place.

Kidney stones with complete blockage.
```

4.4.2 Concept discovery

Manually generated lexical resources such as WordNet (Miller 1990) are extremely useful in applications such as word sense disambiguation and question answering. Their limitations are discussed in Section 4.1. Concept discovery is the process of automatically inducing semantic classes, like those found in WordNet, from textual data. For example, CBC discovers semantic classes such as *countries*, *pastries*, *scientific disciplines* and *sports teams*.

We use a set of words and the contexts in which the set of words tends to occur to define a concept. We introduce some linguistic information in the feature representation of the words. The meaning of an unknown word can often be inferred from its context. Consider the following sentences:

wine		suit	
Beer	0.190	lawsuit	0.317
white wine	0.185	jacket	0.176
red wine	0.185	shirt	0.161
Chardonnay	0.176	pant	0.152
champagne	0.145	dress	0.149
fruit	0.143	case	0.141
food	0.142	sweater	0.133
coffee	0.136	coat	0.130
Juice	0.134	trouser	0.129
Cabernet	0.128	claim	0.128
cognac	0.128	blazer	0.125
vinegar	0.126	slack	0.124
Pinot noir	0.126	business suit	0.124
milk	0.125	blouse	0.122
vodka	0.124	skirt	0.122
grape	0.123	litigation	0.121
Zinfandel	0.122	complaint	0.119
Cabernet Sauvignon	0.122	Jean	0.119
olive oil	0.121	vest	0.119
sauce	0.121	legal action	0.117

Figure 4.4 – Top-20 similar words of *wine* and *suit*.
The similar words were obtained from the TREC corpus using Lin's similarity measure (Lin 1998b).

A bottle of *tezgũno* is on the table.
Everyone likes *tezgũno*.
Tezgũno makes you drunk.
We make *tezgũno* out of corn.

The contexts in which the word *tezgüno* is used suggest that *tezgüno* may be a kind of alcoholic beverage. This is because other alcoholic beverages tend to occur in the same contexts as *tezgüno*. The intuition is that words that occur in the same contexts tend to be similar. This is known as the Distributional Hypothesis (Harris 1985). There have been many approaches to computing the similarity between words based on their distribution in a corpus (Hindle 1990; Lin 1998b). The output of these programs is a ranked list of similar words to each word. For example, Lin's approach outputs the top-20 similar words of *wine* and *suit* shown in Figure 4.4.

Following (Lin 1998b), we represent each word by a feature vector where each feature corresponds to a context in which the word occurs. We use the parser Minipar, described in Section 3.2, to collect the contexts in which words occur. For example, "threaten with __" is a context. If the word *handgun* occurred in this context, then the context is a feature of *handgun*. As in document clustering, the value of the feature is the pointwise mutual information between the feature and the word. These contexts and scores are inserted into the feature database described in Section 3.3.1

As an extension to concept discovery (using the soft-clustering version of Phase III of CBC, see Section 4.2.3), we can discover word senses. In Figure 4.4, the similar words of *wine* represent the meaning of *wine*. However, the similar words of *suit* represent a mixture of its *clothing* and *litigation* senses. Such lists of similar words do not distinguish between the multiple senses of polysemous words (Resnik 1998).

CBC discovers word senses by assigning words to more than one cluster. Each cluster to which a word is assigned represents a sense of that word. Using the *Soft-clustering* version of Phase III of CBC allows CBC to assign words to multiple clusters, to discover the less frequent senses of a word and to avoid discovering duplicate senses.

Chapter 5

Evaluation Methodology

Evaluating clustering results is a very difficult task. We will now review some common approaches to automatically evaluating clustering algorithms. Many of these methodologies lack either generality or goal-orientation. We will propose two novel methodologies that attempt to strike a balance between the two. The first methodology is based on the editing distance between clustering results and manually constructed classes (the answer key). It measures the percentage of savings obtained by using the clustering result to construct the answer key versus constructing it from scratch (i.e. a baseline clustering). The second methodology, specific to word sense discovery, measures the precision and recall of discovered senses using WordNet as the gold standard of senses. We provide a mechanism for mapping a sense discovered by CBC to a WordNet synset and formulate whether the discovered sense is correct.

We conclude by describing an evaluation methodology in which clustering outputs are embedded in language modeling (Goodman 2001). The standard *n*-gram language model is presented as well as the predictive clustering model, which combines *n*-grams and clusters. We then describe perplexity, the standard evaluation measure used in language modeling.

5.1 Previous approaches

Many cluster evaluation schemes have been proposed. They generally fall under two categories:

- comparing cluster outputs with manually generated answer keys (hereon referred to as classes); or
- embedding the clusters in an application (e.g. information retrieval) and using its evaluation measure.

5.1.1 Comparison with manually generated answer keys

An example of the first type of approach considers the average entropy of the clusters, which measures the purity of the clusters (Steinbach, Karypis, Kumar 2000). Given a clustering *C* and an answer class *A*, for each cluster *c* in *C* we compute the class distribution as:

$$p_{ca} = \frac{f(c,a)}{f(c,*)} \qquad \text{(Eq. 5.1)}$$

where *a* is a class in *A*, *f(c, a)* is the number of elements in cluster *c* that intersect with class *a* and the asterix (*) represents a wildcard. This methodology assumes that classes are disjoint and that they contain all elements to be clustered. The entropy of cluster *c* is then:

$$E_c = \sum_{a \in A} p_{ca} \log(p_{ca}) \qquad \text{(Eq. 5.2)}$$

Finally, the overall entropy (or purity) is given by:

$$E = \sum_{c \in C} \frac{|c| \times E_c}{|C|} \qquad \text{(Eq. 5.3)}$$

This measure prefers small tight clusters. In fact, maximum purity, 0, is trivially achieved when each element forms its own cluster.

Another way to evaluate clusters is to compute the percentage of the decisions that are in agreement between the clusters and the classes (Wagstaff and Cardie 2000). Given a partitioned set of *n* elements, there are $n \times (n - 1) / 2$ pairs of elements that are either in the same partition or not. So, a partition implies $n \times (n - 1) / 2$ decisions. The percentage of the decisions that are in agreement between the clusters *C* and the classes *A*, *d*(*C*, *A*) is given by:

$$d(C, A) = \frac{2 \times agreements(C, A)}{n(n-1)} \qquad \text{(Eq. 5.4)}$$

where agreements(*C*, *A*) gives the number of agreements between the clusters *C* and the classes *A*. This measure sometimes gives unintuitive results. Suppose the answer key consists of 20 equally sized classes with 1000 elements in each. Treating each element as its own cluster gets a misleadingly high score of 95%. Consider a slightly better clustering with 2 clusters where each cluster contains half of the classes in the answer key. The resulting score is 55%.

5.1.2 Application-embedded evaluation

An example of a methodology that evaluates clusters by embedding them in an application is in document retrieval (Hearst and Pedersen 1996). Suppose we cluster the documents returned by a search engine. Assuming the user is able to pick the most relevant cluster, the performance of the clustering algorithm can be measured by the average precision of the chosen cluster. Under this scheme, only the best cluster matters.

Another example is in smoothing probability distributions using language modeling (Lee and Pereira 1999). A methodology for this is described in detail in Section 5.4.

5.2 Editing distance

The entropy and pairwise decision schemes each measure a specific property of clusters. However, these properties are not directly related to the application-level goals of clustering. The information retrieval scheme is goal-oriented, however it measures only the quality of the best cluster. We now propose an evaluation methodology that strikes a balance between generality and goal-orientation.

5.2.1 Methodology

Like the entropy and pairwise decision schemes, we assume that there is an answer key that defines how the elements are supposed to be clustered. We also assume a hard clustering where each element belongs to exactly one cluster. Let C be a set of clusters and A be the answer key. We define the editing distance, *dist*(C, A), as the number of operations required to transform C into A. We allow three editing operations:

- *merge* two clusters;
- *move* an element from one cluster to another; and
- *copy* an element from one cluster to another.

The *copy* operation is only needed if the answer classes are not disjoint. Let B be the baseline clustering where each element is its own cluster. We define the quality of a clustering C as follows:

$$quality(C, A, B) = 1 - \frac{dist(C, A)}{dist(B, A)} \quad \text{(Eq. 5.5)}$$

This measure can be interpreted as the percentage of savings obtained by using the clustering result to construct the answer key versus constructing it from scratch (i.e. a baseline clustering). Suppose there are m classes in the answer key. We assume that we start with a list of m empty sets, each of which is labelled with a class in the answer key. The goal is to use the editing operations to make the sets identical to the answer class A. The transformation procedure is as follows:

1) For each cluster, merge it with the set whose class has the largest number of elements in the cluster (a tie is broken arbitrarily).
2) If an element is in a set whose class is not the same as one of the element's classes, move the element to a set where it belongs.
3) If an element belongs to more than one target class, copy the element to all sets corresponding to the target classes (except the one to which it already belongs).

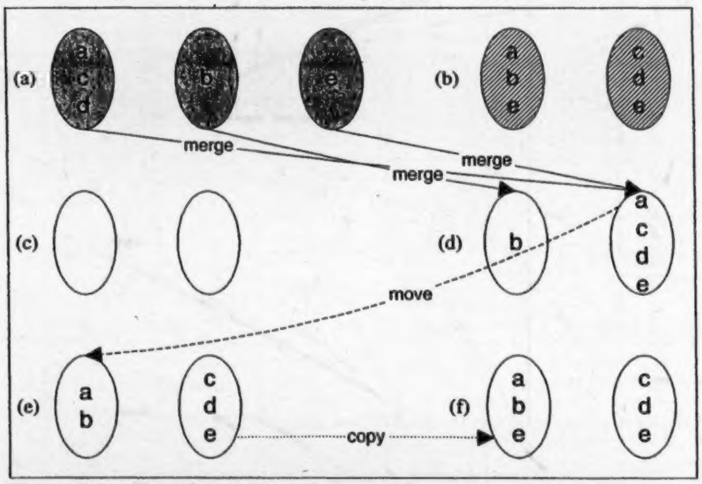


Figure 5.1 – Transformation rules example.
(a) The clusters to be transformed; (b) the classes in the answer key; (c) the sets used to reconstruct the classes (Initialization); (d) the sets after three merge operations (Step 1); (e) the sets after one move operation (Step 2); (f) the sets after one copy operation (Step 3).

dist(*C*, *A*) is the number of operations performed using the above transformation rules on *C*. Figure 5.1 shows an example. In (d) the cluster containing *e* could have been merged with either set (we arbitrarily chose the second). The total number of operations is five.

Proposition: The above transformation procedure generates the optimal number of operations for any clustering.

Proof: Suppose there are *n* clusters. Since we have a hard clustering, each cluster contains an element that must be merged into the sets. Therefore, each cluster must be merged at least once with a set. The transformation procedure produces exactly *n merge* operations, which is the minimum number possible. If the answer class contains *p* duplicate elements then at least *p copy* operations are required since we have a hard clustering[7]. The transformation procedure generates exactly *p copy* operations. Now, all that there is left to show is that the optimal number of *move* operations is performed by the transformation procedure. The *copy* operations do not influence the number of *move* operations, only the *merge* operations do. Let *c* be a cluster and s_1 be the set whose class has the largest number of elements in *c*. Suppose that in Step 1, *c* is merged with a set s_2 whose class does not have the largest number of elements in the cluster *c*. Let $f(c, s)$ be the number of elements that intersect between cluster *c* and set *s*. $f(c, s_1) > f(c, s_2)$. Therefore, more

---

[7] Note that it is only assumed that the clustering is hard. The answer class may in fact contain duplicate elements. This scenario will be observed in our concept discovery application.

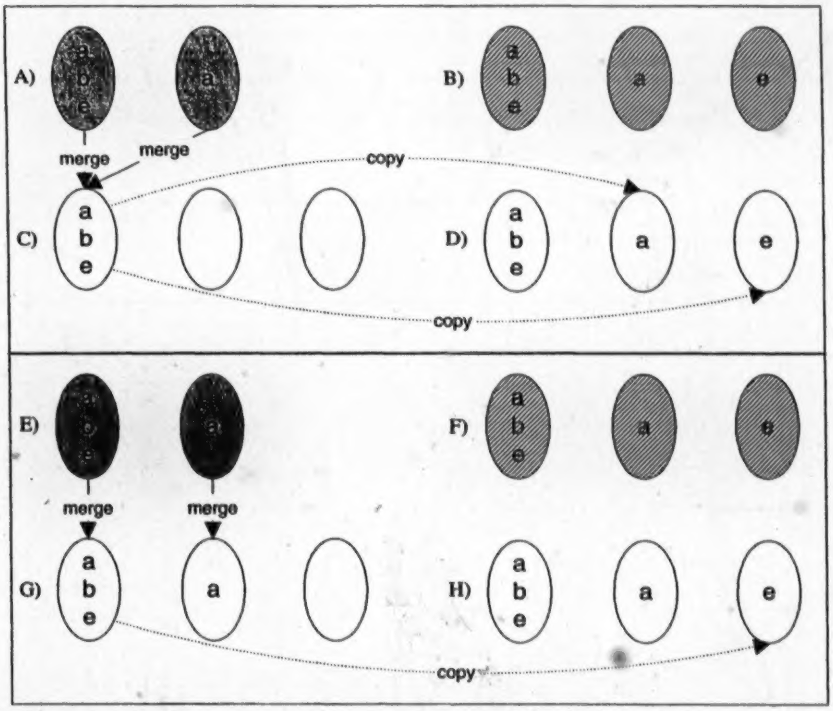


Figure 5.2 – Counter-example to transformation rules' optimality for soft clusterings.
The upper rectangle shows the four editing operations performed by the transformation rules while the lower rectangle shows that the transformation is possible using three editing operations. (a) and (e) The clusters to be transformed; (b) and (f) the classes in the answer key; (c) and (g) the sets after two *merge* operations; (d) the sets after two *copy* operations; (h) the sets after one *copy* operation.

move operations (exactly $f(c, s_1) - f(c, s_2)$) will be required if c is initially merged with s_2. Now, could there be fewer moves if Step 1 remains the same but then more merges are performed? Suppose in Step 1, cluster c_1 is merged with set s_1 and cluster c_2 is merged with set s_2. Then, suppose we merge s_1 into s_2. The elements of c_1 that belong to s_1 will now have to be moved or copied from s_2 into s_1. But, since $f(c_1, s_1) \geq f(c_1, s_2)$ the number of moves and copies cannot be less then before the merge. Thus, at least one extra operation will occur with the extra *merge* operation. Hence, the number of operations performed by the transformation procedure is optimal. ■

Now, suppose that we have a soft clustering. The optimality claim no longer holds. The transformation rules may even use varying numbers of editing operations depending on its random choices in Step 1. Consider the example in Figure 5.2. The Steps (a) – (d) illustrate the

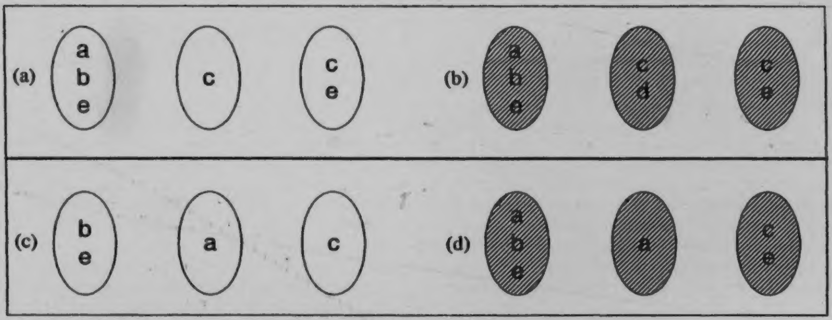


Figure 5.3 – Two examples of sets consistent with classes.
(a) and (c) are sets consistent with the classes in (b) and (d) respectively.

four editing operations performed by the transformation rules when a particular random choice is made in Step 1. (e) – (h) shows a transformation sequence that results in only three editing operations if the other choice in Step 1 was made.

5.2.2 Consistent extension

In some clustering applications, it is not reasonable to transform a clustering to be identical to the answer key *A*. For example, in concept discovery from textual data, some senses in *A* may not exist in the corpus used to generate *C*. In this dissertation, we will extract answer classes from WordNet for our concept discovery algorithm. In WordNet, the word *dog* belongs to both the *Person* and *Animal* classes. However, in the newspaper corpus that we use to discover concepts, the *Person* sense of *dog* is at best extremely rare. There is no reason to expect a clustering algorithm to discover this sense of *dog*.

For such data, we redefine the editing distance, *dist*(*C*, *A*), as the number of operations required to make *C* consistent with *A*. We say that *C* is consistent with *A* if there is a one to one mapping between the clusters in *C* and the classes in *A* such that for each cluster *c* in *C*, all elements of *c* belong to the same class in *A*. Figure 5.3 illustrates an example. We now only allow the first two editing operations since the *copy* operation is no longer needed. The baseline distance *dist*(*B*, *A*) is now exactly the number of elements to be clustered since each element *e* only needs to be merged with the set that corresponds to a class that contains *e*.

5.3 Evaluating word senses

The second evaluation methodology we propose is used to evaluate our word sense discovery extension to concept discovery. Below, we describe a method of obtaining the precision, recall and *F*-measure of a sense discovery output. We provide a mechanism for mapping a sense discovered by CBC to a WordNet synset and formulate whether the discovered sense is correct.

5.3.1 Precision

For each word in the sense discovery application, CBC outputs a list of clusters to which the word belongs (i.e. a soft clustering using Version 2 of Phase III of CBC). The goal is that each cluster corresponds to a sense of the word. The **precision** of a word measures the percentage of output clusters that actually correspond to a sense of the word. To compute this precision, we must define what it means for a cluster to correspond to a correct sense of a word. Consider the following output of CBC:

```
(palm
   Nq19     0.21 (fir, douglas fir, pine, eucalyptus)
   Nq570    0.14 (towel, cheesecloth, sponge, rag)
   Nq3      0.12 (abdomen, thigh, buttock, wrist)
)
```

Intuitively, Nq19 and Nq3 correspond to two correct senses of *palm* while Nq570 does not. To automatically determine whether a cluster corresponds to a correct sense of a word *w*, we map the cluster to a WordNet sense of *w*.

Let *S*(*w*) be the set of WordNet senses of a word *w* (each sense is a synset that contains *w*). We define *simW*(*s*, *u*), the similarity between a synset *s* and a word *u*, as the maximum similarity between *s* and a sense of *u*:

$$simW(s,u) = \max_{t \in S(u)} sim(s,t) \qquad \text{(Eq. 5.6)}$$

where *sim*(*s*, *t*) is defined in Eq. 3.1 on page 34.

Let c_k be the top-k members of a cluster c, where these are the k most similar members to the committee of c. We define the similarity between s and c, $simC(s, c)$, as the average similarity between s and the top-k members of c:

$$simC(s,c) = \frac{\sum_{u \in c_k} simW(s,u)}{k} \qquad \text{(Eq. 5.7)}$$

Suppose a clustering algorithm assigns the word w to a cluster c. We say that c corresponds to a **correct sense** of w if the maximal average similarity between a synset containing w and the top-k members of c exceeds a threshold θ:

$$\max_{s \in S(w)} simC(s,c) \geq \theta \qquad \text{(Eq. 5.8)}$$

In our experiments, we set k = 4 and varied the θ values. The WordNet sense of w that corresponds to c is then:

$$\arg\max_{s \in S(w)} simC(s,c) \qquad \text{(Eq. 5.9)}$$

It is possible that multiple clusters to which w is assigned will correspond to the same WordNet sense. In this case, we only automatically count one of them as correct.

We define the **precision** of a word w as the percentage of correct clusters to which it is assigned. The precision of a clustering algorithm is the average precision of all the words.

In the *palm* example, using θ = 0.25, Nq19 correctly maps to the *tree* sense of *palm* in WordNet with average similarity 0.644. The sense of *palm* that is most similar to *towel*, *cheesecloth*, *sponge* and *rag* is the *palm/body_part*. However, the average similarity is only 0.093, which is less than the threshold 0.25. Therefore, Nq570 is incorrect. Nq3 correctly maps to *palm/body_part* with average similarity 0.364. The precision of *palm* is 2/3. Figure 6.5, in our experimental results, shows the general effect of varying the value of θ on the clustering output of several algorithms.

5.3.2 Recall

The **recall** (completeness) of a word *w* measures the ratio between the correct clusters to which *w* is assigned and the actual number of senses in which *w* was used in the corpus. Clearly, there is no way to know the complete list of senses of a word in any non-trivial corpus. To address this problem, we pool the results of several clustering algorithms to construct the target senses. For a given word *w*, we use the union of the correct clusters of *w* discovered by a set of clustering algorithms as the target list of senses for *w*. While this recall value is likely not the true recall, it does provide a relative ranking of the algorithms used to construct the pool of target senses.

Suppose another sense discovery algorithm outputs the following clusters for *palm*:

```
(palm
   C1 0.23 (pine, oak, fir, redwood)
   C2 0.10 (inch, meter, foot, yard)
)
```

The clusters C1 and C2 are correctly mapped to *palm_tree* and *palm/linear_unit* senses of *palm*, respectively. The target senses for *palm* from the two algorithms are: *palm_tree*, *palm/body_part* and *palm/linear_unit*. Therefore, the recall of *palm* is 2/3 for both algorithms.

The overall recall is the average recall of all words.

5.3.3 F-measure

It is sometimes useful to have a single measure that combines precision and recall aspects. One such measure is the *F*-measure (Shaw Jr., Burgin and Howell 1997), which is the harmonic mean of recall and precision:

$$F = \frac{1}{\alpha\frac{1}{R} + (1-\alpha)\frac{1}{P}} \qquad \text{(Eq. 5.10)}$$

where R is the recall and P is the precision. Typically, $\alpha = ½$ is used:

$$F = \frac{2RP}{R+P} \qquad \text{(Eq. 5.11)}$$

F weighs low values of precision and recall more heavily than higher values. It is high when both precision and recall are high.

5.4 Language Modeling

As part of our evaluation of our concept discovery system, we will embed concepts in a language modeling task. The goal of language modeling is to predict sequences of words. Language models are key in applications such as speech recognition (Church 1988), machine translation (Brown et al. 1990), language identification (Dunning 1994), optical character recognition (OCR) (Hull and Srihari 1982), and spelling correction (Kernighan, Church and Gale 1990). In this section, we describe the *n*-gram language model and the standard perplexity evaluation metric. We proceed by describing predictive clustering, which is a language model that incorporates clusters with *n*-grams.

5.4.1 Review

Given all the words previously seen in some text, a language model estimates the probability of the next word:

$$P(w_i \mid w_1, w_2, \dots, w_{i-1}) \qquad \text{(Eq. 5.12)}$$

where w_k is the k^{th} word in the text. For large *i*, it becomes difficult to compute this probability. A typical estimation is to assume that w_i is dependent only on the previous $n - 1$ words:

$$P(w_i \mid w_1, w_2, \dots, w_{i-1}) \approx P(w_i \mid w_{i-n+1}, \dots, w_{i-1}) \qquad \text{(Eq. 5.13)}$$

This model is referred to as an *n*-gram model. In a bigram model ($n = 2$), the current word is dependant only on its previous word whereas, in a trigram model ($n = 3$), it depends on its two previous words. *n*-gram models are very effective for language modeling. A typical problem encountered when using an *n*-gram model is zero probabilities. Suppose we use a trigram model to estimate the probability of a document. Many trigrams in the document will never have been seen before and consequently, Eq. 5.13 will yield a zero probability. This is referred to as the data sparseness problem. Smoothing algorithms such as Backoff (Katz 1987) and deleted interpolation

(Jelinek and Mercer 1980) reserve probability mass for unseen events to avoid generating zero probability events.

Several techniques have been developed to improve upon the *n*-gram model: skipping, caching, sentence mixtures, and clustering (Goodman 2001). In skipping models, instead of using the previous $n - 1$ words as history, we condition on a displaced context. For example, instead of computing $P(w_i \mid w_{i-2}, w_{i-1})$, we may compute $P(w_i \mid w_{i-3}, w_{i-2})$. In a caching model, it is assumed that the probability of a word increases if we have seen it recently. In a sentence mixture model, it is assumed that there are different types of sentences (e.g. general news story, financial statement, sports stat lines, etc.) and that these are then modelled separately. Each of these models, on its own, often performs worse than the standard *n*-gram model. They are typically used in combination with the *n*-gram model or even with each other.

5.4.2 Perplexity

Perplexity measures how well a model predicts some data. It is the standard measure for comparing the predictability of different language models over a corpus. From information theory, the entropy of a random variable X, $H(X)$, with probability mass function $p(x)$, is defined as:

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x) \qquad \text{(Eq. 5.14)}$$

X describes what we are predicting (e.g. words in language modeling). The *logs* calculated may be in any base. Using base 2 provides us with an intuitive description of entropy: it is a lower bound on the number of bits required to optimally encode X. A related measure, *cross-entropy*, is useful in comparing an unknown probability mass function $p(x)$ with an approximation $m(x)$. Cross-entropy, $H(p, m)$, is defined as:

$$H(p,m) = -\sum_{x \in X} p(x) \log_2 m(x) \qquad \text{(Eq. 5.15)}$$

Cross-entropy gives us an upper bound on the true entropy of X, $H(p, m) \geq H(X)$. But, we do not know $p(x)$. A model X is stationary if $p(x)$ does not change over time and ergodic if $p(x)$ is aperiodic and irreducible. If X is stationary and ergodic, then the cross-entropy may be simplified to (Cover and Thomas 1991):

$$H(p,m) = \lim_{n \to \infty} -\frac{1}{n} \log_2 m(x_1 x_2 x_3 \ldots x_n) \qquad \text{(Eq. 5.16)}$$

Now, we can obtain an upper bound on *H*(*X*) without knowing *p*(*x*). For finite strings, we only get an approximation to *H*(*p*, *m*). A related measure, *cross-perplexity*, is defined as:

$$perplexity(p,m) = 2^{H(p,m)} \qquad \text{(Eq. 5.17)}$$

To be consistent with the literature, we now refer to this measure as simply perplexity, although cross-perplexity is the correct term. Roughly speaking, the perplexity of a random variable *X* measures the number of weighted decisions that must be made by the model *X*. If *X* predicts words, then perplexity measures the number of weighted choices we have for a word at any point in time.

Now, let *W* be a random variable describing words. The true probability mass function, *p*(*w*), is unknown. Let us define *m*(*w*), an approximation to *p*(*w*), as an *n*-gram model. Since Markov models such as *n*-gram models are stationary and ergodic, the simplified cross-entropy formula applies. Then, the cross-entropy of our model may be computed as:

$$H(p,m) = \lim_{n \to \infty} -\frac{1}{n} \log_2 m(w_1 w_2 w_3 \ldots w_n) \qquad \text{(Eq. 5.18)}$$

Using a testing corpus *L* with a finite number of words *n*, we approximate *H*(*p*, *m*) with:

$$H(p,m) = -\frac{1}{n} \sum_{w \in L} \log_2 m(w) \qquad \text{(Eq. 5.19)}$$

where *w* is a word in *L*. The better the language model, the more predictive the model will be of a testing corpus (i.e. the lower *H*(*p*, *m*) will be). Given two language models and a training corpus, we have the probability mass function $m_1(w)$ for the first model and $m_2(w)$ for the second model. After computing the perplexity of each probability mass function on a same test set, we say that the first language model is better than the second if:

$$H(p,m_1) < H(p,m_2) \qquad \text{(Eq. 5.20)}$$

or

$$2^{H(p,m_1)} < 2^{H(p,m_2)} \quad \text{(Eq. 5.21)}$$

5.4.3 Predictive Clustering

Clusters may be used in language models to relieve the data sparseness. Suppose we have seen strings like *drink Pepsi* and *drink cola*. Even though *drink Sprite* may never have been seen, we may hypothesize that *Sprite* can indeed follow *drink* since *Sprite* is very similar to *Pepsi* and *cola*. Let W_i be the cluster that contains word w_i. Predictive clustering is a language model that incorporates clusters as follows (Goodman and Gao 2000):

$$\begin{aligned} P(w_i \mid w_{i-n+1},\ldots,w_{i-1}) = {} & P(W_i \mid w_{i-n+1},\ldots,w_{i-1}) \\ & \times P(w_i \mid w_{i-n+1},\ldots,w_{i-1},W_i) \end{aligned} \quad \text{(Eq. 5.22)}$$

Suppose we have a trigram model and that *BEVERAGE* is the cluster that contains the word *Sprite*. Then, predictive clustering computes the probability of *Sprite* in the string *He drinks Sprite* as:

$$P(Sprite \mid He\,drinks) = P(BEVERAGE \mid He\,drinks) \times P(Sprite \mid He\,drinks\,BEVERAGE)$$

It is important to note that the above equality is true only with a hard clustering. That is, each element must belong to only one cluster.

Given a clustering, the probability model of Eq. 5.22 may be estimated using a training corpus. For a set of *s* clusterings, we may generate *s* language models and compare them using the perplexity measure described in the previous section. A better clustering will generate a language model with lower perplexity than the other language models.

Chapter 6

Experimental Results

We experimented with CBC in the document clustering and concept discovery tasks described in Section 4.4. In this chapter, we describe the data sets used for document clustering and concept discovery and we apply the evaluation methodologies presented in the previous chapter. We will compare CBC's results with those of the clustering algorithms reviewed in Chapter 2 and we will inspect a sample of CBC's outputs.

6.1 Document clustering

In this section, the test data used in our document clustering system is first described. We then apply the editing distance evaluation methodology from Section 5.2.1 to the clustering outputs obtained from CBC and the algorithms from Chapter 2. We proceed by evaluating the robustness of CBC by studying the effect of some of its clustering parameters on clustering output. Finally, we present an illustrative example of a CBC output on a small data set consisting of the research papers presented at the 2001 SIGIR conference.

Table 6.1 – Document clustering test data sets.
The number of classes in each test data set and the number of elements in their largest and smallest classes.

DATA SET	*TOTAL DOCS*	*TOTAL CLASSES*	*LARGEST CLASS*	*SMALLEST CLASS*
Reuters	2745	92	1045	1
20-news	18828	20	999	628

6.1.1 Experimental setup

We conducted document clustering experiments with two data sets: Reuters-21578 V1.2[8] and 20news-18828[9]. Table 6.1 describes for each data set the total number of documents, the number of classes, and the number of elements in their largest and smallest classes. For the Reuters corpus, we selected the 2745 documents that:

1) are assigned one or more topics;

2) have the attribute LEWISSPLIT="TEST"; and

3) have <BODY> and </BODY> tags.

The 20news-18828 data set contains 18,828 newsgroup articles partitioned nearly evenly across 20 different newsgroups. We will now use Reuters to refer to the Reuters-21578 V1.2 data set and 20-news to refer to the 20news-18828 data set.

6.1.2 Cluster evaluation

We clustered the data sets using CBC and the clustering algorithms of Chapter 2 and applied the evaluation methodology from Section 5.2.1. Table 6.2 shows the results. The columns are our editing distance-based evaluation measure. On the 20-news data set, our implementation of Chameleon was unable to complete in reasonable time. For *K*-means, we used *K*=1000 over five iterations for 20-news and *K*=50 over eight iterations for Reuters. For Buckshot, we used *K*=80 for 20-news and *K*=50 for Reuters, each over eight iterations. For the 20-news corpus, CBC

---

[8] Available at http://www.research.att.com/~lewis/reuters21578.html.

[9] Available at http://www.ai.mit.edu/people/jrennie/20_newsgroups/.

Table 6.2 – Cluster quality (%) of various clustering algorithms on document clustering.
The quality is computed using the Reuters and 20-news data sets.

	REUTERS	*20-NEWS*
CBC	**65.00**	**74.18**
K-means	62.38	70.04
Buckshot	62.03	65.96
Bisecting K-means	60.80	58.52
Chameleon	58.67	n/a
Average-link	63.00	70.43
Complete-link	46.22	64.23
Single-link	31.53	5.30

spends the vast majority of the time finding the top similar documents (38 minutes) and computing the similarity between documents and committee centroids (119 minutes). The rest of the computation, which includes clustering the top-20 similar documents for every one of the 18828 documents and sorting the clusters, took less than 5 minutes. We used a Pentium III 750MHz processor and 1GB of memory. CBC outperforms each algorithm. Its closest competitors were average-link and *K*-means, which CBC beat by 3.75% and 4.14% respectively. The algorithms performed better on the 20-news corpus because that corpus contains much fewer classes (20 vs. 92) and many more examples (18,828 vs. 2745).

Buckshot has similar performance to *K*-means on the Reuters corpus; however it performs much worse on the 20-news corpus (see Table 6.2). This is because *K*-means performs well on this data set when *K* is large (e.g. *K*=1000) whereas Buckshot cannot have *K* higher than 137. On the Reuters corpus, the best clusters for *K*-means were obtained with $K = 50$, and Buckshot can have *K* as large as $\sqrt{2745} = 52$. However, as *K* approaches 52, Buckshot degenerates to the *K*-means algorithm, which explains why Buckshot has similar performance to *K*-means.

6.1.3 Clustering parameters

CBC has several parameters that can affect its clustering output. We experimented with several of them. Below, we describe each parameter and its possible values:

1) **Vector Space Model**: (described in Section 3.3.2) This is the feature representation for the elements (e.g. documents) to be clustered. In the mutual-information model, the statistic for each element-feature pair is the pointwise mutual information between the element and the feature. In the term-frequency model, the statistic is simply the frequency of the element-feature pair. The term-frequency / inverse-document frequency (*tf-idf*) model combines the term-frequency model with the inverse of the percentage of documents in which the term occurs. The possible values of this parameter are:

   a) MI : the mutual-information model;

   b) TF : the term-frequency model, *tf*;

   c) TFIDF1 : the *tf-idf* model: $tf\text{-}idf = tf \times log\ idf$;

   d) TFIDF2 : the *tf-idf* model using the formula: $tf\text{-}idf = \sqrt{tf} \times log\ idf$.

2) **Stemming**: Stemming is the process of removing the common morphological and inflexional endings from words in English (described in Section 4.4.1). We use Porter's stemmer (Porter 1980). The possible values of this parameter are:

   a) `S-` : terms are not stemmed;

   b) `S+` : terms are stemmed using Porter's stemmer.

3) **Stop Words**: Stop words are frequently occurring common words such as *be*, *at*, *a*, *with*, *for*, etc. These words are often insignificant and ignored. Our list of stop words contains about 200 words. The possible values of this parameter are:

   a) `W-` : no stop words are used as features;

   b) `W+` : all terms are used.

4) **Filtering**: Filtering is used to reduce the size of the feature vectors, making the similarity computations much faster, as described in Section 4.2.1.

   a) `F-` : no term filtering is performed;

   b) `F+` : terms with $MI<0.5$ are deleted.

We refer to an experiment using a string where the first position corresponds to the *Stemming* parameter, the second position corresponds to the *Stop Words* parameter and the third position

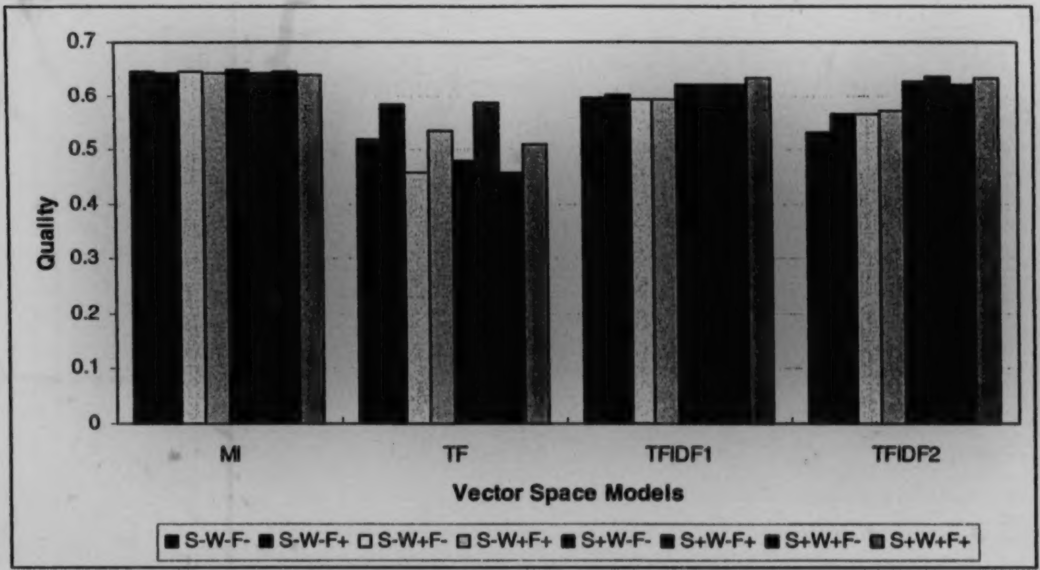


Figure 6.1 – CBC evaluation of cluster quality when varying clustering parameters. Cluster quality is obtained using the Reuters corpus.

corresponds to the *Filtering* parameter. For example, experiment `S+W-F+` means that terms are stemmed, stop words are ignored, and filtering is performed. The *Vector Space Model* parameter will always be explicitly given.

Figure 6.1 illustrates the quality of the clusters generated by CBC on the Reuters corpus while varying the clustering parameters. Most document clustering systems use *TFIDF1* as their vector space model; however, the *MI* model outperforms each model including *TFIDF1*. Furthermore, varying the other parameters on the *MI* model makes no significant difference on cluster quality, making CBC with *MI* more robust. *TF* performs the worst since terms with low discriminating power (e.g. *the* and *furthermore*) are not discounted. Although *TFIDF2* slightly outperforms *TFIDF1* on experiments `S+W-F-` and `S+W-F+`, it is clearly not as robust. Except for the *TF* model, stemming terms always produced better quality clusters.

6.1.4 Illustrative example

The results shown in Table 6.2 show that CBC outperforms the other clustering algorithms. However, by simply looking at the numbers in that table, it is difficult to get an appreciation for the quality of the clustering. When clustering words, we can inspect a sample of the clustering to

Table 6.3 – Output of CBC applied to the 46 papers presented at SIGIR-2001.
The left column shows the clustered documents and the right column shows the top-7 features (stemmed terms) forming the cluster centroids.

#	CLUSTER ELEMENTS	TOP-7 FEATURES (STEMMED)
1	Cat/017, Cat/019, Lrn/037	text, featur, learn, categor, classif, approach, svm
2	MS/035, Eval/011, MS/036, Sys/006, Cat/018	threshold, score, term, base, distribut, optim, scheme
3	LM/015, LM/041, LM/016, CL/012, CL/013, CL/014	model, languag, translat, expans, estim, improv, framework
4	US/029, US/028, US/027, Sys/008, Sum/024, Eval/009, Lrn/038	user, result, use, imag, system, search, index
5	Web/030, Sys/007, MS/033, Eval/010	search, engin, page, web, link, best
6	Sum/002, Lrn/039, LM/042	stori, new, event, time, process, applic, content
7	Sum/003, Sum/004, Sum/025, LM/043, Sum/001	summar, term, text, summari, weight, scheme, automat
8	Lrn/040, Sys/005	level, space, framework, vector, comput, recent
9	QA/046, QA/044, QA/045, MS/034	answer, question, perform, task, passag, larg, give
10	RM/021, RM/022, RM/023, RM/020	queri, languag, similar, data, framework, seri
11	Web/032, Sum/026, Web/031	link, algorithm, method, hyperlink, web, analyz, identifi

Cat=Categorization, CL=CrossLingual, Eval=Evaluation, LM=LanguageModels, Lrn=Learning, MS=MetaSearch, QA=QuestionAnswering, RM=RetrievalModels, Sum=Summarization, Sys=Systems, US=UserStudies, Web=Web

get an intuitive idea of its quality. This is because humans have prior knowledge about the meaning of words. For document clustering, we would have to supplement a sample of the clustering with the content of the documents clustered, which is too large. For our illustrative example, we therefore used a smaller data set.

We collected the titles and abstracts for the 46 papers presented at SIGIR-2001 and clustered them using CBC. For each paper, we used as part of its filename the session name in which it was presented at the conference and a number representing the order in which it appears in the proceedings. For example, Cat/017 refers to a paper that was presented in the *Categorization* session and that was the 17th paper in the proceedings. The resulting 11 clusters generated by CBC are shown in Table 6.3.

The features of many of the automatically generated clusters clearly correspond to SIGIR-2001 session topics (e.g. clusters 1, 4, 7, 9, 10 and 11). Applying the editing distance evaluation methodology of Section 5.2.1 gives a quality of 32.60%. This score is fairly low for the following reasons:

- Some documents could potentially belong to more than one session. For example, Lrn/037 was clustered in the categorization cluster #1 because it deals with learning and text categorization (it is titled "A meta-learning approach for text categorization"). Using the sessions as the answer key, Lrn/037 will be counted as incorrect.

- CBC generates clusters that do not correspond to any session topic. For example, all papers in Cluster #6 have news stories as their application domain and the papers in Cluster #5 all deal with search engines.

6.2 Concept discovery

In this section, we describe our experimental setup and present the results of evaluating CBC when it is used in concept discovery. To apply the editing distance evaluation methodology from Section 5.2, we need a test set and an answer class. However, unlike in document clustering, a standard test set does not exist for concept discovery. We therefore show how to construct one using the senses in WordNet. It is then possible to compare the outputs of CBC and the algorithms from Chapter 2 using the editing distance methodology.

To get an intuitive idea of the quality of CBC outputs we will analyze the quality of a random sample of concepts. We then proceed with evaluating the clustering algorithms by embedding them in predictive clustering language models and computing their relative perplexity, the standard evaluation used in language modeling, described in Section 5.4.2. We will show that the relative rankings of the clustering algorithms are similar for the language modeling evaluation and our editing distance evaluation on both document clustering and concept discovery.

Finally, we will experiment with the word sense discovery extension to concept discovery described in Section 4.4.2. We will apply the precision/recall methodology from Section 5.3 to the word senses generated by CBC and a subset of the clustering algorithms from Chapter 2. We will study the effect of two important thresholds in word sense discovery: the θ threshold (Eq. 5.8) that defines a correct sense mapping to a WordNet synset and the σ threshold that determines how many senses a word will have. We will also perform a manual evaluation of a random sample of CBC concepts and compare our results with the automatic evaluation. We then conclude by inspecting the quality of several low-precision clusters.

Table 6.4 – Concept discovery test data sets.
Properties of the test sets used for evaluating cluster quality of the concept discovery task.

DATA SET	TOTAL WORDS	M	AVG. FEATURES PER WORD	TOTAL CLASSES
S_{13403}	13403	250	740.8	202
S_{3566}	3566	3500	2218.3	150

6.2.1 Experimental setup

To extract target classes from WordNet, we first estimate the probability of a random word belonging to a subhierarchy (a synset and its hyponyms). As described in Section 3.1, we use the frequency counts of synsets in the SemCor corpus to get the estimated probabilities. Figure 3.1 illustrates an example WordNet hierarchy of synsets with their probabilities. A class is then defined as a maximal subhierarchy with probability less than a threshold (we used e^{-2}).

We used Minipar, from Section 3.2, to parse about 1GB (144M words) of newspaper text from the TREC collection (1988 AP Newswire, 1989-90 LA Times, and 1991 San Jose Mercury) at a speed of about 500 words/second on a PIII-750 with 512MB memory. We collected the frequency counts of the grammatical relationships (contexts) output by Minipar and used them to compute the pointwise mutual information values from Section 3.3.2. The test set is constructed by intersecting the words in WordNet with the nouns in the corpus whose total mutual information with all of its contexts exceeds a threshold m. Since WordNet has a low coverage of proper names, we removed all capitalized nouns. We constructed two test sets: S_{13403} consisting of 13,403 words ($m = 250$) and S_{3566} consisting of 3566 words ($m = 3500$). We then removed from the answer classes the words that did not occur in the test sets. Table 6.4 summarizes the test sets. The sizes of the WordNet answer classes vary considerably. For S_{13403} there are 99 classes that contain three words or less and the largest class contains 3246 words. For S_{3566}, 78 classes have three or less words and the largest class contains 1181 words.

6.2.2 Cluster evaluation

We clustered the test sets using CBC and the clustering algorithms of Chapter 2 and applied the evaluation methodology from Section 5.2. Table 6.5 shows the results. The columns are our

Table 6.5 – Cluster quality (%) of several clustering algorithms on S_{3566} and S_{13403}.

ALGORITHM	S_{13403}	S_{3566}
CBC	**60.95**	**65.82**
K-means (*K*=250)	56.70	62.48
Buckshot	56.26	63.15
Bisecting *K*-means	43.44	61.10
Chameleon	n/a	60.82
Average-link	56.26	62.62
Complete-link	49.80	60.29
Single-link	20.00	31.74

editing distance-based evaluation measure. Test set S_{3566} has a higher score for all algorithms because it has a higher number of average features per word than S_{13403}.

For the *K*-means and Buckshot algorithms, we set the number of clusters (*K*) to 250 and the maximum number of iterations to 8. We used a sample size of 2000 for Buckshot. For the Bisecting *K*-means algorithm, we applied the basic *K*-means algorithm twice ($\alpha = 2$ in Section 2.2.2) with a maximum of 8 iterations per split. Our implementation of Chameleon was unable to complete clustering S_{13403} in reasonable time due to its time complexity. *K*-means, Buckshot and average-link have very similar performance. CBC outperforms all algorithms on both data sets.

The relative ranking of the clustering algorithms according to cluster quality is nearly identical to that reported in our evaluation of document clustering in Table 6.2. This supports the editing distance evaluation methodology presented in Section 5.2. Figure 6.2 plots the cluster quality of both the document clustering test set 20-news, described in Section 6.1.1, and S_{13403} over several clustering algorithms. The shapes are nearly identical except for at the single-link point. On the 20-news test set, our implementation of single-link grouped every element into one large cluster.

6.2.3 Manual Inspection

Let *c* be a cluster and *wn*(*c*) be the WordNet class that has the largest intersection with *c*. The **precision** of *c* is defined as:

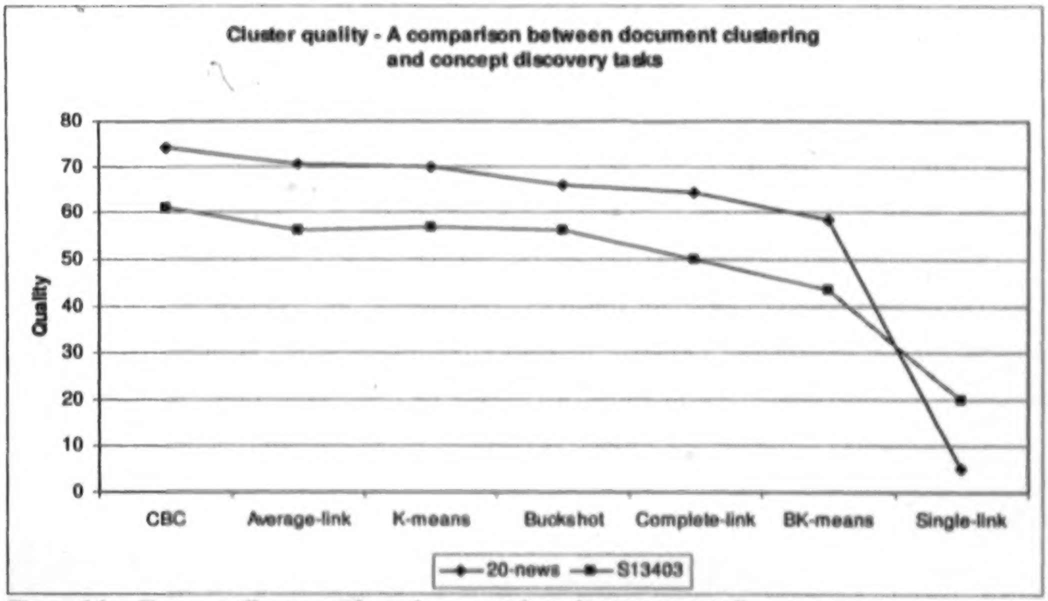


Figure 6.2 – Cluster quality comparison: document clustering vs. concept discovery.
The cluster quality of the 20-news document clustering test set and the S_{13403} concept discovery test set plotted over several clustering algorithms.

$$precision(c) = \frac{|c \cap wn(c)|}{|c|} \qquad \text{(Eq. 6.1)}$$

Note that this measure of precision is different from the one presented in Section 5.3.1. That one will be used in our evaluation of word sense discovery in Section 6.2.6.

CBC discovered 943 clusters from S_{13403}. We sorted them according to their precision using Eq. 6.1. Table 6.6 shows five of the clusters evenly distributed according to their precision ranking along with their Top-15 features with highest mutual-information. The words in the clusters are listed in descending order of their similarity to the cluster centroid. For each cluster *c*, we also include *wn*(*c*). The underlined words are in *wn*(*c*).

The first cluster is clearly a cluster of firearms and the second is one of pests. In WordNet, the word *pest* is curiously only under the *person* hierarchy. The words *stopwatch* and *houseplant* do not belong to the clusters but they have low similarity to their cluster centroid. The third cluster represents some kind of control. In WordNet, the legal power sense of *jurisdiction* is not a hyponym of *social control* as are *supervision*, *oversight* and *governance*. The fourth cluster is about mixtures. The words *blend* and *mix* as the event of mixing are present in WordNet but not

Table 6.6 – Manual inspection of five of the 943 clusters discovered by CBC from S_{13403}. Columns include each cluster's members and features with top-15 highest mutual information as well as the WordNet classes that have the largest intersection with each cluster. The rank is the placement of the clusters in the precision-sorted list of all clusters.

RANK	*MEMBERS*	*TOP-15 FEATURES*	*WN(C)*
1	handgun, revolver, shotgun, pistol, rifle, machine gun, sawed-off shotgun, submachine gun, gun, automatic pistol, automatic rifle, firearm, carbine, ammunition, magnum, cartridge, automatic, stopwatch	__ blast, barrel of __ , brandish __, fire __, point __, pull out __, __ discharge, __ fire, __ go off, arm with __, fire with __, kill with __, open fire with __, shoot with __, threaten with __	artifact / artefact
236	whitefly, pest, aphid, fruit fly, termite, mosquito, cockroach, flea, beetle, killer bee, maggot, predator, mite, houseplant, cricket	__ control, __ infestation, __ larvae, __ population, infestation of __, specie of __, swarm of __ , attract __, breed __, eat __, eradicate __, feed on __, get rid of __, repel __, ward off __	animal / animate being / beast / brute / creature / fauna
471	supervision, discipline, oversight, control, governance, decision making, jurisdiction	breakdown in __, lack of __ , loss of __, assume __, exercise __, exert __, maintain __, retain __, seize __, tighten __, bring under __, operate under __, place under __, put under __, remain under __	act / human action / human activity
706	blend, mix, mixture, combination, juxtaposition, combine, amalgam, sprinkle, synthesis, hybrid, melange	dip in __, marinate in __, pour in __, stir in __, use in __, add to __, pour __, stir __, curious __, eclectic __, ethnic __, odd __, potent __, unique __, unusual __	group / grouping
941	employee, client, patient, applicant, tenant, individual, participant, renter, volunteer, recipient, caller, internee, enrollee, giver	benefit for __, care for __, housing for __, benefit to __, service to __, filed by __, paid by __, use by __, provide for __, require for --, give to __, offer to __, provide to __, disgruntled __, indigent __	worker

as the result of mixing. The last cluster is about consumers. Here is the *consumer* class in WordNet:

> `addict, alcoholic, big spender, buyer, client, concert-goer, consumer, customer, cutter, diner, drinker, drug addict, drug user, drunk, eater, feeder, fungi, head, heroin addict, home buyer, junkie, junky, lush, nonsmoker, patron, policyholder, purchaser, reader, regular, shopper, smoker, spender, subscriber, sucker, taker, user, vegetarian, wearer`

In our cluster, only the word *client* belongs to WordNet's *consumer* class. The cluster is ranked very low because WordNet failed to consider words like *patient*, *tenant* and *renter* as consumers. Table 6.6 shows that even low ranking CBC clusters are fairly coherent.

Table 6.7 – A comparison of clusters representing the *cell* concept.
The comparison is for several clustering algorithms using S_{13403}.

ALGORITHMS	CLUSTERS WITH THE LARGEST INTERSECTION WITH THE WORDNET CELL CLASS.
CBC	white blood cell, red blood cell, brain cell, cell, blood cell, cancer cell, nerve cell, embryo, neuron
K-means	cadaver, meteorite, secretion, receptor, serum, handwriting, cancer cell, thyroid, body part, hemoglobin, red blood cell, nerve cell, urine, gene, chromosome, embryo, plasma, heart valve, saliva, ovary, white blood cell, intestine, lymph node, sperm, heart, colon, cell, blood, bowel, brain cell, central nervous system, spinal cord, blood cell, cornea, bladder, prostate, semen, brain, spleen, organ, nervous system, pancreas, tissue, marrow, liver, lung, marrow, kidney
Buckshot	cadaver, vagina, meteorite, human body, secretion, lining, handwriting, cancer cell, womb, vein, bloodstream, body part, eyesight, polyp, coronary artery, thyroid, membrane, red blood cell, plasma, gene, gland, embryo, saliva, nerve cell, chromosome, skin, white blood cell, ovary, sperm, uterus, blood, intestine, heart, spinal cord, cell, bowel, colon, blood vessel, lymph node, brain cell, central nervous system, blood cell, semen, cornea, prostate, organ, brain, bladder, spleen, nervous system, tissue, pancreas, marrow, liver, lung, bone marrow, kidney
Bisecting *K*-means	picket line, police academy, sphere of influence, bloodstream, trance, sandbox, downtown, mountain, camera, boutique, kitchen sink, kiln, embassy, cellblock, voting booth, drawer, cell, skylight, bookcase, cupboard, ballpark, roof, stadium, clubhouse, tub, bathtub, classroom, toilet, kitchen, bathroom,
WordNet Class	blood cell, brain cell, cancer cell, cell, cone, egg, nerve cell, neuron, red blood cell, rod, sperm, white blood cell

Table 6.7 shows the clusters containing the word *cell* that are discovered by various clustering algorithms from S_{13403}. The underlined words represent the words that belong to the *cell* class in WordNet. The CBC cluster corresponds almost exactly to WordNet's *cell* class. *K*-means and Buckshot produced fairly coherent clusters. The cluster constructed by Bisecting *K*-means is obviously of inferior quality. This is consistent with the fact that Bisecting *K*-means has a much lower score on S_{13403} compared to CBC, *K*-means and Buckshot.

6.2.4 Language Modeling

We embedded the outputs obtained in Section 6.2.2 for several clustering algorithms into predictive clustering language modeling, described in Section 5.4.3. The algorithms we used were CBC, *K*-means, Buckshot, Bisecting *K*-means, and average-link. The predictive clustering models were trained using the 1991 San Jose Mercury files, a subset of the TREC corpus. The six training sets extracted from these files are described in Table 6.8.

Table 6.8 – Language modeling training sets.
The training sets are drawn from the 1991 San Jose Mercury corpus.

TRAINING SET	*FILE NAMES*	*SIZE (MB)*	*WORDS*
T_5	SJM_001 – SJM_005	5	589,724
T_{10}	SJM_001 – SJM_010	10	1,143,800
T_{25}	SJM_001 – SJM_025	25	2,871,660
T_{50}	SJM_001 – SJM_050	50	5,710,310
T_{75}	SJM_001 – SJM_075	75	8,544,940
T_{100}	SJM_001 – SJM_100	100	11,368,700

Each predictive clustering model uses trigrams and Katz' backoff for smoothing for zero-probability events (Katz 1987). We employed the standard trigram language model with Katz' backoff as a baseline for our evaluation. We computed the perplexity (see Section 5.4.2) of each language model over each training set. The perplexity is computed using a testing set consisting of the SJM_298 and SJM_299 files (about 219, 454 words).

Figure 6.3 shows the relative perplexity of each predictive clustering model with the standard trigram model. Each curve represents a language model and each point represents the difference in perplexity between that point's model and the standard trigram model. The lower the perplexity, the better the model.

All algorithms converge towards the standard trigram model as more training data is available. This is expected since clusters are most useful when data is sparse. As additional training data is available, more trigrams in the test set will have been seen during training. With smaller training sets, the quality of clustering is more apparent. All algorithms, apart from CBC, have higher perplexity than the standard trigram model. A poor clustering smoothes across unrelated words and produces higher perplexity whereas a good clustering smoothes across related (or similar) words and lowers perplexity.

Both the training sets and testing set are taken from the 1991 San Jose Mercury files. We applied the same methodology to a comparable (i.e. news story domain) but different test set consisting of files AP881206 and AP881209 from the 1988 AP Newswire corpus (approximately 244,606 words). The training data remains the same. Figure 6.4 shows the results. The shapes of the curves are similar to those in Figure 6.3 and the relative rankings are also the same. However, there is a larger gap in relative perplexity with smaller training data. There are more unseen

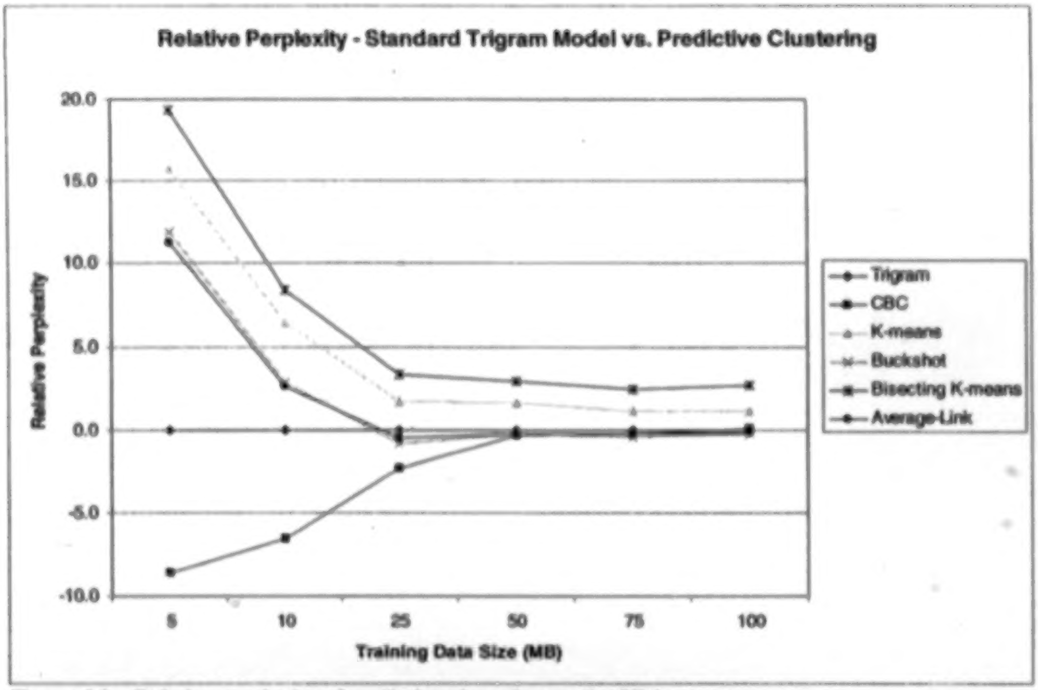


Figure 6.3 – Relative perplexity of predictive clustering on the SJM test set.
The predictive clustering language models use CBC, *K*-means, Buckshot, Bisecting *K*-means, and average-link clustering. Each point represents the difference between that model's perplexity and the standard trigram model's perplexity. Katz' backoff smoothing is used for all models.

trigrams in the AP test set so the clusters are more important than in the SJM test set. The absolute values of perplexity are significantly worse on the AP test set: 413 for AP vs. 194 for SJM.

The relative ranking of the clustering algorithms in Figure 6.3 and Figure 6.4 remains the same as those reported in Table 6.5 for S_{3566} and similar to those reported in Table 6.2.

6.2.5 Sample concepts

Appendix A lists 30 of the concepts discovered by CBC on the ACQUAINT corpus, which is three times larger than the TREC corpus described in Section 6.1.1. It consists of roughly 375 million words correlating to about 3 GB of data. The text is drawn from three newswire sources: Xinhua News Service (People's Republic of China) (1996-2000), the New York Times News Service (1998-2000), and the Associated Press Worldstream News Service (1998-2000). The

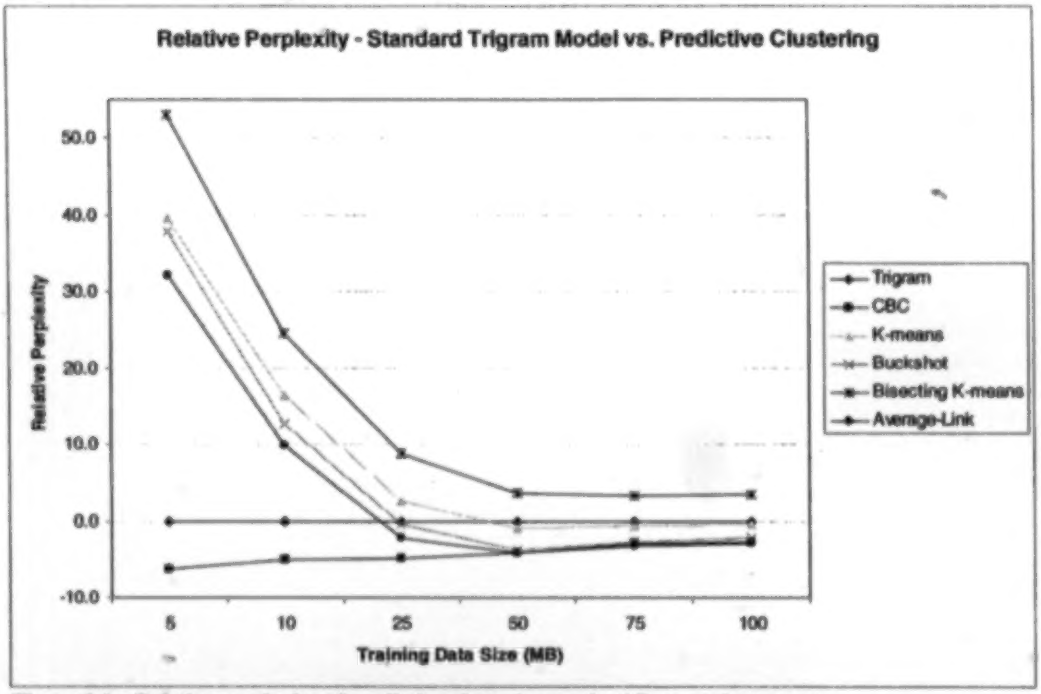


Figure 6.4 – Relative perplexity of predictive clustering on the AP newswire test set.
The predictive clustering language models use CBC, *K*-means, Buckshot, Bisecting *K*-means, and average-link clustering. Each point represents the difference between that models perplexity and the standard trigram model's perplexity. Katz' backoff smoothing is used for all models.

sample shown in Appendix A includes only the words that have a similarity ≥ 0.15 with the centroid of a cluster (i.e. $\sigma = 0.15$ for the algorithm presented in Section 4.2.3).

6.2.6 Word sense discovery

Experimental setup

We used the same 1GB TREC collection as described in Section 6.1.1. We applied the precision/recall evaluation methodology described in Section 5.3 using the test set consisting of 13,403 words, S_{13403}.

We modified the average-link, *K*-means, Bisecting *K*-means and Buckshot algorithms reviewed in Chapter 2 since these algorithms only assign each element to a single cluster. For each of these algorithms, the modification is as follows:

Table 6.9 – Precision, Recall and *F*-measure on S_{13403} for various algorithms. The evaluation parameters are $\sigma = 0.18$ and $\theta = 0.25$.

ALGORITHM	*PRECISION (%)*	*RECALL (%)*	*F-MEASURE (%)*
CBC	**60.8**	**50.8**	**55.4**
UNICON	53.3	45.5	49.2
Buckshot	52.6	45.2	48.6
K-means	48.0	44.2	46.0
Bisecting K-means	33.8	31.8	32.8
Average-link	50.0	41.0	45.0

```
Apply the algorithm as described in Chapter 2
For each cluster c returned by the algorithm
   Create a centroid for c using all elements assigned to it
Apply MK-means using the above centroids
```

where *MK*-means is the *K*-means algorithm, using the above centroids as initial centroids, except that each element is assigned to its most similar cluster plus all other clusters with which it has similarity greater than σ. We then use these modified algorithms to discover senses.

These clustering algorithms were not designed for sense discovery. Like UNICON, when assigning an element to a cluster, they do not remove the overlapping features from the element. Thus, a word is often assigned to multiple clusters that are similar. Also, as discussed in Section 4.2.3, an infrequent sense of a word, like the life sense of *plant*, will likely not be discovered.

Word Sense Evaluation

We ran CBC and the modified clustering algorithms described above on the data set S_{13403} and applied the precision/recall evaluation methodology from Section 5.3. Appendix B lists a 1% random sample of the polysemous words discovered by CBC. Table 6.9 shows the results of our experiment. For Buckshot and *K*-means, we set the number of clusters to 250 and the maximum number of iterations to 8. For the Bisecting *K*-means algorithm, we applied the basic *K*-means algorithm twice ($\alpha = 2$) with a maximum of 8 iterations per split. CBC returned 943 clusters and outperformed the next best algorithm by 7.5% on precision and 5.3% on recall.

In Section 5.3.1, we stated that a cluster corresponds to a correct sense of a word *w* if its maximum *simC* similarity with any synset in *S*(*w*) exceeds a threshold θ (Eq. 5.8). Figure 6.5

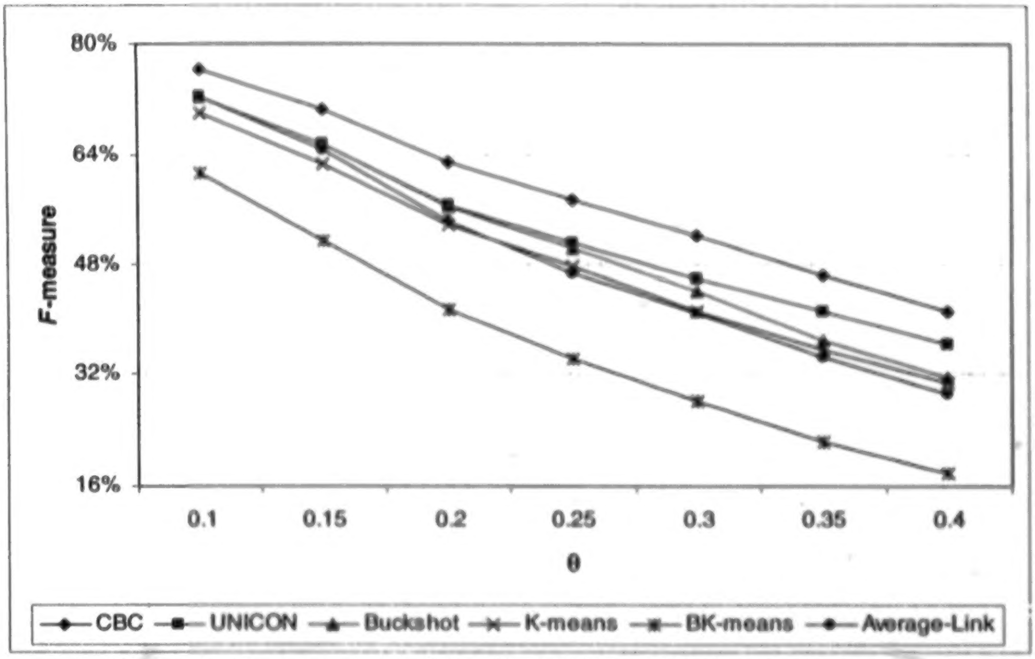


Figure 6.5 – *F*-measure of several algorithms with σ = 0.18 and varying θ thresholds.

shows our experiments using different values of θ. The higher the θ value, the stricter we are in defining correct senses. Naturally, the systems' *F*-measures decrease when θ increases. The relative ranking of the algorithms is not sensitive to the choice of θ values. CBC has higher *F*-measure for all θ thresholds.

For all sense discovery algorithms, we assign an element to a cluster if their similarity exceeds a threshold σ. The value of σ does not affect the first sense returned by the algorithms for each word because each word is always assigned to its most similar cluster. We experimented with different values of σ and present the results in Figure 6.6. With a lower σ value, words are assigned to more clusters. Consequently, the precision decreases while recall increases. CBC has higher *F*-measure for all σ thresholds.

Manual Evaluation

We manually evaluated a 1% random sample of the test data consisting of 133 words with 168 senses. Here is an example of the instances that would be manually judged for the words *aria*, *capital* and *device*:

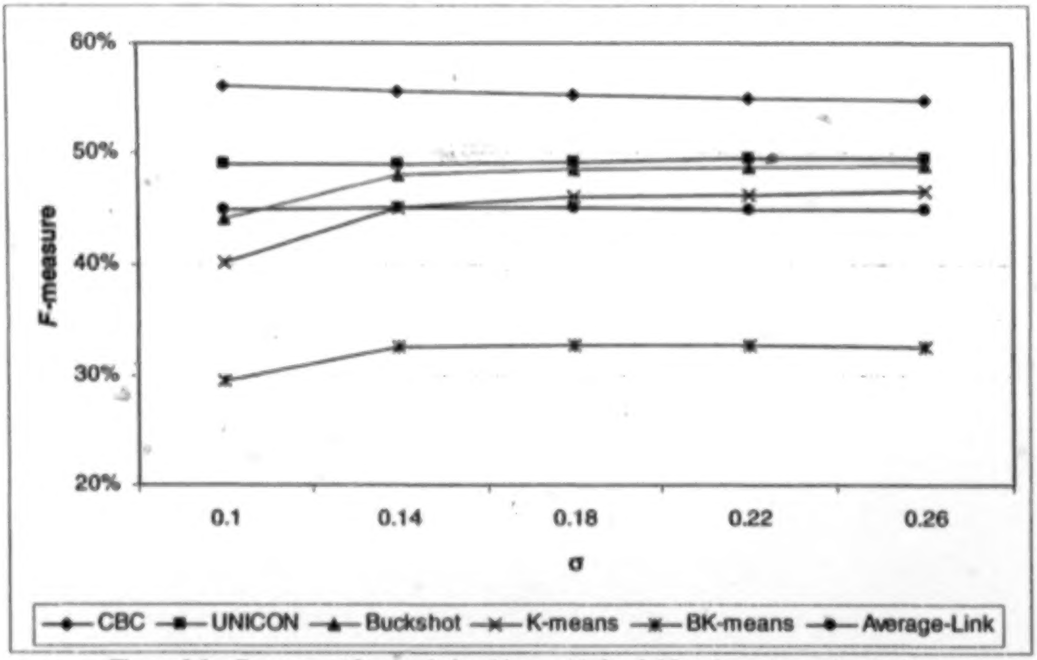


Figure 6.6 – *F*-measure of several algorithms with $\theta = 0.25$ and varying σ thresholds.

aria	S_1:	song, ballad, folk song, tune
capital	S_1:	money, donation, funding, honorarium
capital	S_2:	camp, shantytown, township, slum
device	S_1:	camera, transmitter, sensor, electronic device
device	S_2:	equipment, test equipment, microcomputer, video equipment

For each discovered sense of a word, we include its top-4 most similar words. Appendix C lists all the instances that were manually judged. The evaluation consists of assigning a tag to each sense as follows:

- √: The list of top-4 words describes a sense of the word that has not yet been seen
- ×: The list of top-4 words does not describe a sense of the word
- +: The list of top-4 words describes a sense of the word that has already been seen (duplicate sense)

The S_2 sense of *device* is an example of a sense that is evaluated with the duplicate sense tag.

Table 6.10 – Confusion matrix of manual vs. automatic evaluations.
The comparison is of a 1% random sample of the data set.

		MANUAL EVALUATION		
		√	×	+
Automatic	√	**104**	2	0
	×	17	**41**	0
	+	0	1	**3**

Table 6.10 compares the agreements/disagreements between our manual and automatic evaluations. Our manual evaluation agreed with the automatic evaluation 88.1% of the time. This suggests that the evaluation methodology is reliable.

Most of the disagreements (17 out of 20) were on senses that were incorrect according to the automatic evaluation but correct in the manual evaluation. The automatic evaluation misclassified these because sometimes WordNet misses a sense of a word and because of the organization of the WordNet hierarchy. Some words in WordNet should have high similarity (e.g. *elected official* and *legislator*) but they are not close to each other in the hierarchy.

Our manual evaluation of the sample gave a precision of 72.0%. The automatic evaluation of the same sample gave 63.1% precision. Of the 13,403 words in the test data, CBC found 2869 of them polysemous.

Discussion

We computed the average precision for each cluster, which is the percentage of elements in a cluster that correctly correspond to a WordNet sense according to Eq. 5.8. We inspected the low-precision clusters and found that they were low for three main reasons.

First, some clusters suffer from part-of-speech confusion. Many of the nouns in our data set can also be used as verbs and adjectives. Since the feature vector of a word is constructed from all instances of that word (including its noun, verb and adjective usage), CBC outputs contain clusters of verbs and adjectives. For example, the following cluster contains 112 adjectives:

```
weird, stupid, silly, old, bad, simple, normal, wrong,
wild, good, romantic, tough, special, small, real, smart,
...
```

The noun senses of all of these words in WordNet are not similar. Therefore, the cluster has a very low 2.6% precision. In hindsight, we should have removed the verb and adjective usage features.

Secondly, CBC outputs some clusters of proper nouns. If a word that first occurs as a common noun also has a proper-noun usage it will not be removed from the test data. For the same reasons as the part-of-speech confusion problem, CBC discovers proper noun clusters but gets them evaluated as if they were common nouns (since WordNet contains few proper nouns). For example, the following cluster has an average precision of 10%:

> blue jay, expo, angel, mariner, cub, brave, pirate, twin, athletics, brewer

Finally, some concepts discovered by CBC are completely missing from WordNet. For example, the following cluster of government departments has a low precision of 3.3% because WordNet does not have a synset that subsumes these words:

> public works, city planning, forestry, finance, tourism, agriculture, health, affair, social welfare, transport, labor, communication, environment, immigration, public service, transportation, urban planning, fishery, aviation, telecommunication, mental health, procurement, intelligence, custom, higher education, recreation, preservation, lottery, correction, scouting

Somewhat surprisingly, all of the low-precision clusters that we inspected are reasonably good. At first sight, we thought the following cluster was bad:

> shamrock, nestle, dart, partnership, haft, consortium, blockbuster, whirlpool, delta, hallmark, rosewood, odyssey, bass, forte, cascade, citadel, metropolitan, hooker

Here are some of the features of the centroid of this cluster:

> __ product, __ customer, work at __, leave __, __ announced, __ disclosed, __ introduced, etc.

By looking at the features, we realized that it is mostly a cluster of company names.

Chapter 7

Conclusions

We proposed a general-purpose clustering algorithm called CBC (Clustering By Committee) from which we organized documents according to topics and from which we discovered concepts and word senses. We explored the value of these systems by experimenting with two novel evaluation methodologies that defined what a word sense is and defined the quality of a clustering output.

7.1 Contributions

The main contribution of this dissertation is the general purpose clustering algorithm CBC. It addresses the general goal of clustering, which is to group data elements such that the intra-group similarities are high and the inter-group similarities are low. Using sets of representative elements called committees, CBC attempts to discover cluster centroids that unambiguously describe the members of a possible class. The algorithm initially discovers committees that are well scattered in the similarity space. It then proceeds by assigning elements to their most similar clusters. After assigning an element to a cluster, CBC removes their overlapping features from the element before assigning it to another cluster. This allows CBC to discover the less frequent senses of a word and to avoid discovering duplicate senses.

We explored CBC's use in learning from textual data by applying it to the tasks of document clustering and concept discovery. Document clustering is practical in many information retrieval tasks such as document browsing and the organization and viewing of retrieval results. Apart from the CBC algorithm itself, the concept and word sense discovery system is the next largest contribution from this work as it may become an important resource in the natural language processing and data mining communities. Broad-coverage lexical resources such as WordNet are extremely useful but are mostly hand generated. They often include many rare senses while missing domain-specific senses. Automatically generating them is useful for many applications such as word sense disambiguation, question answering and ontology construction. CBC was shown to outperform several common clustering strategies in both document clustering and concept discovery tasks.

Evaluating clustering results has remained a very challenging task. We presented two novel evaluation methodologies to automatically evaluate clustering output. These methodologies provided measures that are more intuitive and easier to interpret than previous measures. The first is based on the editing distance between output clusters and a manually constructed answer key. It defines how much work is necessary in order to convert from one to the other. Intuitively, it determines the percentage of savings of using the clustering result to construct an answer key versus constructing it from scratch (i.e. a baseline clustering). The experiments conducted using various testing sets and by a separate evaluation using language modeling resulted in similar rankings of the clustering algorithms. CBC consistently outperformed all other algorithms.

The second methodology is specific to word sense discovery. It measures the precision and recall of the senses discovered by CBC using WordNet as the gold standard of senses. We provided a mechanism for mapping a sense discovered by CBC to a WordNet synset and formulated whether the discovered sense is a correct sense. In our experiments, CBC surpassed the next best algorithm by 7.5% on precision and 5.3% on recall. A manual evaluation of a 1% sample of the concepts discovered by CBC agreed 88.1% of the time with the automatic evaluation supporting the reliability of the evaluation methodology. 85% of the disagreements were on senses that were incorrect according to the automatic evaluation but correct in the manual evaluation. These disagreements were mostly due to lackings in the WordNet hierarchy.

We studied the effect of various CBC parameters on clustering quality. Using the mutual-information vector space model, CBC is robust to design choices such as stemming terms, removing stop words, filtering, and the minimum similarity threshold for assigning elements to clusters in our soft clustering model. We also showed that CBC generates reasonable clusters even when using a minuscule corpus such as the 46 papers presented at SIGIR-2001. Finally, in a manual inspection of the concepts and senses generated from the TREC corpus, we showed that even the lowest ranking CBC clusters are fairly coherent.

7.2 Future work

CBC is already being used by several research groups in the world. One group is looking at clustering French words. Without a French parser, the feature representation must be modified from our concept discovery application. A simple approach is to use proximity features. This approach assumes that words can be characterized by the words that surround them. Instead of representing each word by the contexts in which it occurs, we may represent each word by the words that surround it. Consider the following sentence:

> `Edmonton won the seventh` **`game`** `at home, and then went 11-0 last year en route to their fourth Cup in five seasons.`

In our concept discovery application, the word *game* in this sentence would have the following features: *-V:obj:N:win*, *N:det:D:the*, and *N:adj:A:seventh*. Using a proximity window of two words, the features of *game* would be *-2:the*, *-1:seventh*, *+1:at*, and *+2:home*. Using a similarity measure like (Lin 1998b) to generate a similarity matrix of words using proximity features yields comparable results to using dependency features. It is therefore expected that reasonable results would be obtained by applying CBC to a French corpus using proximity features.

CBC may also be used in word sense disambiguation (WSD). The goal of WSD is to resolve the ambiguity of polysemous words in text. For example, the following two sentences depict two different senses of the word *chair*:

> `If I threw 500` **`chairs`** `in 900 games, that would be one thing, but I've thrown one and now I'm the` **`chair-thrower`**`.`

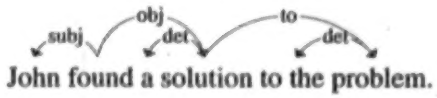
> `Support of those candidates is not dependent on whether they support me for caucus` **`chair`**`.`

Typical WSD systems disambiguate a word by tagging it with a WordNet synset. A standard hand-coded testing corpus called SemCor (Miller, Chodorow, Landes and Leacock 1994) is usually used for evaluating systems. CBC offers the capability of disambiguating senses in an unsupervised manner. Given a polysemous word *w* in a parsed sentence, we can use *w*'s local features to assign it to an existing concept. Using Eq. 5.9, we can map this concept to a WordNet synset and assign that sense to *w*.

A group is looking at CBC's discovered concepts as a first step in identifying a large number of words (particularly polysemous words) for an automatically created ontology representing complex entities that have different perspectives (e.g. a *chair* can be something to sit on or the head of a group; an *ensemble* can be a musical group or a clothing outfit). Another group is looking at clustering web pages. Here, the bag-of-words model might not be sufficient for representing small pages. Additional features that include the topology of the website in which the page exists may help.

In earlier work, we presented an algorithm, called DIRT (Discovery of Inference Rules from Text), to automatically learn paraphrase expressions from text (Lin and Pantel 2001b). It is a generalization of previous algorithms used for finding similar words (Hindle 1990; Pereira, Tishby and Lee 1993; Lin 1998b). These algorithms use the Distributional Hypothesis, which states that words that occurred in the same contexts tend to have similar meanings (Harris 1985). Instead of applying the Distributional Hypothesis to words, we applied it to paths in dependency trees. Essentially, if two paths tend to link the same sets of words, we hypothesized that their meanings are similar. A path is an expression that represents a binary relationship between two nouns and we generated an inference rule for each pair of similar paths. For example, Table 7.1 lists the 50 most similar paths to "*X solves Y*" generated by DIRT.

Consider the following parse tree for the sentence *John found a solution to the problem*:



An example of a path extracted by DIRT for this sentence is between *John* and *problem*: N:subj:V←find→V:obj:N→solution→N:to:N. The left and right sides (*X* and *Y*) of the path are called slot fillers. For this particular sentence, the path has two fillers (features): *X* = *John* and *Y* = *problem*. Using a large corpus, we can collect all the paths along with their features into a feature database as described in Section 3.3.1. CBC may then be directly applied using the database.

CBC may also be useful in question answering. Falcon (Pasca and Harabagiu 2001), one of the top question answering systems, uses an answer type taxonomy that is linked to WordNet

Table 7.1 – Top-50 most similar paths to "*X solves Y*" generated by DIRT.

1.	Y is solved by X	1.	X clears up Y
2.	X resolves Y	2.	*X creates Y
3.	X finds a solution to Y	3.	*Y leads to X
4.	X tries to solve Y	4.	Y is eased between X
5.	X deals with Y	5.	X gets down to Y
6.	Y is resolved by X	6.	X worsens Y
7.	X addresses Y	7.	X ends Y
8.	X seeks a solution to Y	8.	*X blames something for Y
9.	X do something about Y	9.	X bridges Y
10.	X solution to Y	10.	X averts Y
11.	Y is resolved in X	11.	*X talks about Y
12.	Y is solved through X	12.	X grapples with Y
13.	X rectifies Y	13.	*X leads to Y
14.	X copes with Y	14.	X avoids Y
15.	X overcomes Y	15.	X solves Y problem
16.	X eases Y	16.	X combats Y
17.	X tackles Y	17.	X handles Y
18.	X alleviates Y	18.	X faces Y
19.	X corrects Y	19.	X eliminates Y
20.	X is a solution to Y	20.	Y is settled by X
21.	X makes Y worse	21.	*X thinks about Y
22.	X irons out Y	22.	X comes up with a solution to Y
23.	*Y is blamed for X	23.	X offers a solution to Y
24.	X wrestles with Y	24.	X helps somebody solve Y
25.	X comes to grip with Y	25.	*Y is put behind X

synsets. Given a query, the system identifies an answer type (i.e. a specific semantic category) and prefers answers contained in the WordNet synsets linked to that type. One serious limitation of WordNet in this context is that it does not contain many proper nouns. For example, the *composer* synset in WordNet contains names like Mozart and Beethoven, but it is more illustrative than a complete listing. CBC concepts do not have this limitation. CBC also has the advantage that its concepts are specific to the domain of the text used to generate them. This is useful for domain specific question answering systems.

Text contains a wealth of knowledge about who we are, what we know, how we think, and how we communicate. We are just beginning to tap into the information that is available in the tales we read to our children, the narratives that capture our thoughts, and the stories that shape our world. In this dissertation, we presented some recent advances in automatically acquiring knowledge from text. It is, however, just a drop in the bucket of the vast information available in text.

Bibliography

Agrawal, R.; Gehrke, J.; Gunopulos, D.; and Raghavan, P. 1998. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of ACM SIGMOD International Conference on Management of Data 1998*. 94–105. Seattle, Washington.

Aho, A. V.; Sethi, R.; and Ullman, J. D. 1986. *Compilers – Principles, Techniques and Tools.* Addison-Wesley.

Anderberg, M. R. 1973. *Cluster Analysis for* Applications. Academic Press.

Ankerst, M; Breunig, M.M.; Kriegel, H.-P.; and Sander, J. 1999. OPTICS: Ordering Points To Identify the Clustering Structure. In *Proceedings of ACM SIGMOD International Conference on Management of Data 1999*. 49–60. Philadelphia, PA.

Baeza-Yates, R. A. 1992. Introduction to data structures and algorithms related to information retrieval. In Frakes, W. B. and Baeza-Yates, R. A. (Eds.) *Information Retrieval: Data Structures and Algorithms*. Prentice Hall. pp. 13–27.

Berwick, R. C. 1991. Principles of principle-based parsing. In Berwick, B. C.; Abney, S. P.; and Tenny, C. (Eds.), *Principle-Based Parsing Computation and Psycholinguistics*. pp. 1–38. Kluwer Academic Publishers.

Brown, P. F.; Cocke, J.; Pietra, S. A. D.; Pietra, V. J. D.; Jelinek, F.; Lafferty, J. D.; Mercer, R. L.; and Roosin, P. S. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2): 79–85.

Buckley, C. and Lewit, A. F. 1985. Optimization of inverted vector searches. In *Proceedings of SIGIR-85*. pp. 97–110. Montreal, Canada.

Charniak, E. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL-00*. pp. 132–139. Seattle, WA.

Church, K. 1988. *Phonological Parsing in Speech Recognition*. Kluwer.

Church, K. and Hanks, P. 1989. Word association norms, mutual information, and lexicography. In *Proceedings of ACL-89*. pp. 76–83. Vancouver, Canada.

Collins, M. J. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of ACL-96*. pp. 184–191. Santa Cruz, CA.

Cover, T. M. and Thomas, J. A. 1991. *Elements of Information Theory*. Wiley, New York.

Cutting, D. R.; Karger, D.; Pedersen, J.; and Tukey, J. W. 1992. Scatter/Gather: A cluster-based approach to browsing large document collections. In *Proceedings of SIGIR-92*. pp. 318–329. Copenhagen, Denmark.

Dunning, T. 1994. Statistical identication of language. *Technical Report 94-273*, Computing Research Laboratory, New Mexico State University.

Ester, M.; Kriegel, H.-P.; Sander, J.; and Xu, X. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. 226–231. Portland, OR.

Frakes, W. B. and Baeza-Yates, R. A. 1992. *Information Retrieval: Data Structure and Algorithms*. Prentice Hall.

Goodman. J. 2001. A bit of progress in language modeling. *Computer Speech and Language*, 15:403–434.

Goodman. J. and Gao, J. 2000. Language model compression by predictive clustering. In Proceedings of *International Conference on Spoken Language Processing*. Beijing, China.

Guha, S.; Rastogi, R.; and Shim, K. 1998. Cure: An efficient clustering algorithm for large databases. In *Proceedings of SIGMOD-98*. pp. 73–84. Seattle, WA.

Guha, S.; Rastogi, R.; and Kyuseok, S. 1999. ROCK: A robust clustering algorithm for categorical attributes. In *Proceedings of ICDE'99*. pp. 512–521. Sydney, Australia.

Han, J. and Kamber, M. 2001. *Data Mining – Concepts and Techniques*. Morgan Kaufmann.

Harris, Z. 1985. Distributional structure. In: Katz, J. J. (ed.) *The Philosophy of Linguistics*. New York: Oxford University Press. pp. 26–47.

Hearst, M. A. and Pedersen, J. O. 1996. Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. In *Proceedings of SIGIR-96*. pp. 76–84. Zurich, Switzerland.

Hindle, D. 1990. Noun classification from predicate-argument structures. In *Proceedings of ACL-90*. pp. 268–275. Pittsburgh, PA.

Hull, J. J. and Srihari, S. N. 1982. Experiments in text recognition with binary n-gram and viterbi algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(5):520–530.

Jain, A. K. and Dubes, R. C. 1988. *Algorithms for Clustering Data*. Prentice-Hall.

Jain, A. K.; Murty, M. N.; and Flynn, P. J. 1999. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323.

Jardine, N. and van Rijsbergen, C. J. 1971. The use of hierarchical clustering in information retrieval. *Information Storage and Retreival*, 7:217–240.

Jelinek, F. and Mercer, R. L. 1980. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*. pp. 381–397. Amsterdam, Holland.

Karypis, G.; Han, E.-H.; and Kumar, V. 1999. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer: Special Issue on Data Analysis and Mining*, 32(8):68–75.

Katz, S. 1987. Estimation of probabilities from sparse data for the langauge model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401.

Kaufmann, L. and Rousseeuw, P. J. 1987. Clustering by means of medoids. In Dodge, Y. (Ed.) *Statistical Data Analysis based on the L 1 Norm.* pp. 405–416. Elsevier/North Holland, Amsterdam.

Kaufmann, L. and Rousseeuw, P. J. 1990. *Finding Groups in Data: an Introdution to Cluster Analysis.* John Wiley and Sons.

Kernighan, M.; Church, K.; and Gale, W. A. 1990. A spelling correction program based on a noisy channel model. In *Proceedings of COLING-90.* pp. 205–210. Helsinki, Finland.

King, B. 1967. Step-wise clustering procedures. *Journal of the American Statistical Association,* 69:86–101.

Koller, D. and Sahami, M. 1997. Hierarchically classifying documents using very few words. In *Proceedings of ICML-97.* pp. 170–176. Nashville, TN.

Landes, S.; Leacock, C.; and Tengi, R. I. 1998. Building semantic concordances. In Fellbaum, C. (Ed.) *WordNet: An Electronic Lexical Database.* pp. 199–216. MIT Press.

Leacock, C.; Chodorow, M.; and Miller, G. A. 1998. Using corpus statistics and WordNet relations for sense identification. *Computational Linguistics,* 24(1):147–165.

Lee, L. and Pereira, F. 1999. Distributional similarity models: Clustering vs. nearest neighbors. In *Proceedings of ACL-99.* pp. 33–40. College Park, MD.

Lin, D. 1993. Parsing without overgeneration. In *Proceedings of ACL-93.* pp. 112–120. Columbus, OH.

Lin, D. 1997. Using syntactic dependency as local context to resolve word sense ambiguity. In *Proceedings of ACL-97.* pp. 64–71. Madrid, Spain.

Lin, D. 1998a. Extracting collocations from text corpora. *Workshop on Computational Terminology.* pp. 57–63. Montreal, Canada.

Lin, D. 1998b. Automatic retrieval and clustering of similar words. In *Proceedings of COLING/ACL-98.* pp. 768–774. Montreal, Canada.

Lin, D. 1998c. Dependency-based evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems.* Granada, Spain.

Lin, D. and Pantel, P. 2001a. Induction of semantic classes from natural language text. In *Proceedings of SIGKDD-01.* pp. 317–322. San Francisco, CA.

Lin, D. and Pantel, P. 2001b. Discovery of inference rules for question answering. *Natural Language Engineering* 7(4):343–360.

Lin, D. and Pantel, P. 2002. Concept discovery from text. In *Proceedings of COLING-2002.* pp. 577–583. Taipei, Taiwan.

Manning, C. D. and Schütze, H. 1999. *Foundations of Statistical Natural Language Processing.* MIT Press.

McQueen, J. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematics, Statistics and Probability,* 1:281–298.

Miller, G. 1990. WordNet: An online lexical database. *International Journal of Lexicography,* 3(4).

Miller, G.; Chodorow, M.; Landes, S.; and Leacock, C. 1994. Using a semantic concordance for sense identification. In *Proceedings of the ARPA Human Language Technology Workshop*. pp. 240–243. San Francisco, CA.

Nagy, G. 1968. State of the art in pattern recognition. In *Proceedings of IEEE*, 56(5):836–862.

Pantel, P. and Lin, D. 1998. SpamCop – A spam classification & organization program. In *Proceedings of AAAI-98 Workshop on Learning for Text Categorization*. pp. 95–98. Madison, Wisconsin.

Pantel, P. and Lin, D. 2000. An unsupervised approach to prepositional phrase attachment using contextually similar words. In *Proceedings of ACL-00*. pp. 101–108. Hong Kong.

Pantel, P. and Lin, D. 2002. Discovering word senses from text. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2002*. pp. 613–619. Edmonton, Canada.

Pantel, P. and Lin, D. 2002. Document clustering with committees. In *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval 2002*. pp. 199–206. Tampere, Finland.

Pasca, M. and Harabagiu, S. 2001. The informative role of WordNet in open-domain question answering. In *Proceedings of NAACL-01 Workshop on WordNet and Other Lexical Resources*. pp. 138–143. Pittsburgh, PA.

Pereira, F.; Tishby, N.; and Lee, L. 1993. Distributional clustering of English words. In *Proceedings of ACL-93*. pp. 183–190. Columbus, Ohio.

Porter, M. F. 1980. An algorithm for suffix stripping. In *Proceedings of SIGIR-80*. pp. 318–327. Cambridge, England.

Resnik, P. 1998. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of AI Research*, 11:95–130.

Salton, G. and McGill, M. J. 1983. *Introduction to Modern Information Retrieval*. McGraw Hill.

Sampson, G. 1995. *English for the Computer - The SUSANNE Corpus and Analytic Scheme*. Clarendon Press. Oxford, England.

Shaw Jr., W. M.; Burgin, R.; and Howell, P. 1997. Performance standards and evaluations in IR test collections: Cluster-based retrieval methods. *Information Processing and Management*, 33:1–14.

Sneath, P. H. A. and Sokal, R. R. 1973. *Numerical Taxonomy: The Principles and Practice of Numerical Classification*. Freeman. London, UK.

Steinbach, M.; Karypis, G.; and Kumar, V. 2000. A comparison of document clustering techniques. *Technical Report #00-034*. Department of Computer Science and Engineering, University of Minnesota.

van Rijsbergen, C. J. 1979. *Information Retrieval*, second edition. London: Buttersworth. Available at: http://www.dcs.gla.ac.uk/Keith/Preface.html.

Wagstaff, K. and Cardie, C. 2000. Clustering with instance-level constraints. In *Proceedings of ICML-00*. pp. 1103–1110. Palo Alto, CA.

Zhang, T.; Ramakrishnan, R.; and Livny, M. 1996. BIRCH: An efficient data clustering method for very large databases. In *Proceedings of SIGMOD-96*. pp. 103–113. Montreal, Canada.

Appendices

Appendix A

Sample of 30 concepts discovered by CBC on the 3GB ACQUINT corpus – only words with similarity ≥0.15 with a cluster centroid are listed

Nq0: chili powder, baking powder, paprika, cayenne pepper, curry powder, baking soda, turmeric, allspice, salt, cumin, sugar, cayenne, oregano, nutmeg, pepper, Cinnamon, black pepper, garlic salt, powder, brown sugar, cornstarch, thyme, coriander, Ginger, mustard, fennel seed, cumin seed, white pepper, coriander seed, flour, seasoned salt, vanilla, cardamom, vanilla extract, peppercorn, cornmeal, flake, clove, seasoning, honey, paste, Herb, mustard seed, spice, molasses, saffron, SAGE, yeast, mace, cocoa, cheese

Nq2: Cleveland Indians, Houston Astros, San Diego Padres, Detroit Tigers, Philadelphia Phillies, Pittsburgh Pirates, Cincinnati Reds, Milwaukee Brewers, St. Louis Cardinals, Arizona Diamondbacks, Los Angeles Dodgers, New York Mets, Boston Red Sox, Chicago Cubs, New York Yankees, Atlanta Braves, Baltimore Orioles, Colorado Rockies, Toronto Blue Jays, San Francisco Giants, Texas Rangers, Oakland Athletics, Kansas City Royals, Yankee, Florida Marlins, Anaheim Angels, Chicago White Sox, Seattle Mariners, Tampa Bay Devil Rays, Mets, dodger, Minnesota Twins, Astros, Montreal Expos, Red Sox, brave, Ranger, Giants, White Sox, cub, Padre, diamondback, Orioles, A's, Phillies, mariner, angel, marlin, cardinal, Tampa Bay Lightning, blue jay, Calgary Flames, Washington Capitals, California Angels, devil ray, Indian, Royals, New York Giants, expo, Carolina Hurricanes, red, Columbus Crew, pirate, Toronto Maple Leafs, Yomiuri Giants, Dallas Stars, New York-New Jersey MetroStars, Rockies, Edmonton Oilers, brewer, Sox, Detroit Red Wings, Seibu Lions, Washington Redskins, Boston Bruins, Indianapolis Colts, Ottawa Senators, Colorado Avalanche, Arizona Cardinals, Green Bay Packers, New Jersey Devils, Buffalo Sabres, Los Angeles Kings, Phoenix Coyotes, St. Louis Blues, tiger, Pittsburgh Penguins, Brooklyn Dodgers, Seattle Seahawks, Tampa Bay Mutiny, San Francisco 49ers, Baltimore Ravens, twin, Boston Braves, Washington Senators, Kinder Bologna, ARIZONA, oriole

Nq5: Mike Richter, Tommy Salo, John Vanbiesbrouck, Curtis Joseph, Chris Osgood, Steve Shields, Tom Barrasso, Guy Hebert, Arturs Irbe, Byron Dafoe, Patrick Roy, Bill Ranford, Ed Belfour, Grant Fuhr, Dominik Hasek, Martin Brodeur, Mike Vernon, Ron Tugnutt, Sean Burke, Zach Thornton, Jocelyn Thibault, Kevin Hartman, Felix Potvin, Hasek, Nikolai Khabibulin, Stephane Fiset, Jamie Storr, Olaf Kolzig, Belfour, Roman Turek, Fiset, Salo, Damian Rhodes, Richter, Dan Cloutier, Dafoe, Jorge Campos, Osgood, Hebert, Storr, Barrasso, Kolzig, Gao Hong, Rob Tallas, Peter Schmeichel, Khabibulin, Roy, Brodeur, Briana Scurry, Ranford, Vanbiesbrouck, Walter Zenga, Schmeichel, Grahame, Taffarel, Turek

Nq6: onion, tomato, cucumber, bell pepper, carrot, red onion, eggplant, green onion, scallion, shallot, garlic, celery, potato, zucchini, red bell pepper, red pepper, leek, green pepper, plum tomato, garlic clove, fennel, cilantro, spinach, chilies, water chestnut, radish, bean sprout, cabbage, JALAPENO, beet, turnip, parsnip, bay leaf, green bean, chervil, winter squash, Vidalia onion, Pimento, hard-boiled egg, mushroom, lettuce, asparagus, Basil, chive, pineapple, pancetta, tomatillo, black olive, avocado, sweet potato, snow pea, okra, cantaloupe, chorizo, ARUGULA, artichoke, broccoli, lima bean, plantain, Swiss chard, Cauliflower, brussels sprout, vegetable, lemongrass, lemon, Dill, shiitake mushroom, mango, nectarine, olive, pear, hot pepper, Apple, rhubarb, breadcrumb, TOMATOES, endive, watercress, melon, egg, cherry tomato, chiles, radicchio, Kale, watermelon, mesclun, pumpkin, papaya, bok choy, romaine lettuce, anchovy, fava bean, string bean, peach, green olive, sauerkraut, yuca, Collard, blueberry, kumquat, fig, ROMAINE, ricotta, chard, pinto bean, chili pepper, chili, collard greens, Quince, root vegetable, veggie, plum, BERRY, lime, tofu, passion fruit, Feta, iceberg lettuce, anise, tortellini, sparerib, cranberry, wild rice, matzo, grapefruit, vanilla bean, sweet corn, chutney, persimmon, Tamarind, pomegranate, smoked salmon, salad, corn, YAM, Chinese cabbage, morel, chanterelle, Chipotle, POTATOES, caper, sorrel, kernel, raw meat, ground beef, ORANGE, blackberry, fruit, julienne, truffle, Grand Marnier, licorice, green, corned beef, salsa, tangerine, white wine

Nq10: table tennis, judo, water polo, weightlifting, gymnastics, volleyball, Badminton, archery, field hockey, taekwondo, handball, rowing, soccer, tennis, cycling, track and field, pentathlon, swimming, softball, takraw, wushu, fencing, diving, triathlon, karate, athletics, lacrosse, yachting, billiards, chess, Bowling, swimming event, under-21, snooker, Shooting, croquet, equestrian, squash, three-meter, ice skating, track

Nq12: nausea, dizziness, vomiting, diarrhea, stomachache, pain, fever, headache, skin rash, shortness, constipation, palpitation, fatigue, chest pain, insomnia, symptom, sore throat, upset stomach, memory loss, dehydration, disorientation, drowsiness, numbness, indigestion, jaundice, irritability, tiredness, coughing, cough, irritation, ache, cold, exhaustion, skin disease, panic attack, LETHARGY, weakness, anxiety, bronchitis, seizure, sweating, infection, depression, hot flash, psychosis, rash, Breathing, hallucination

Nq13: computer science, anthropology, sociology, mechanical engineering, zoology, chemical engineering, comparative literature, biology, mathematics, Science, economics, political science, psychology, electrical engineering, Engineering, linguistics, geology, social science, literature, physic, biochemistry, chemistry, physics, liberal arts, math, botany, astronomy, Microbiology, Criminology, astrophysics, molecular biology, geography, agronomy, journalism, pharmacology, physical education, civil engineering, social work, physiology, natural science, theology, fine arts, Archaeology, architecture, environmental science, neuroscience, information science, medicine, English, nursing, civics, communication, earth science, political economy, education, philosophy, accounting, archeology, geophysics, ecology, Oceanography, elementary education, art, paleontology, hydrology, fine art, folklore, humanity, anatomy, FINANCE, language, ethic, theory, algebra, HEALTH, special education, bioethics

Nq14: Cezanne, Gauguin, Renoir, Manet, Matisse, Gogh, Chagall, Monet, Picasso, Degas, van Gogh, Modigliani, Bonnard, Giacometti, Mondrian, Rembrandt, Miro, Kooning, Ingres, Rothko, Vincent van Gogh, Claude Monet, Delacroix, Vermeer, Klee, Jackson Pollock, Pollock, Goya, Van Dyck, Dali, Rodin, Leger, Norman Rockwell, Heade, Andy Warhol, Calder, Warhol, Caravaggio, Salvador Dali, Jasper Johns, Michelangelo

Nq15: multiple sclerosis, diabetes, Parkinson's disease, OSTEOPOROSIS, cardiovascular disease, Parkinson's, rheumatoid arthritis, heart disease, disease, ASTHMA, cancer, hypertension, lupus, high blood pressure, arthritis, emphysema, epilepsy, cystic fibrosis, leukemia, hemophilia, DISORDER, congestive heart failure, Alzheimer, myeloma, glaucoma, schizophrenia, lung cancer, illness, infection, anemia, liver disease, muscular dystrophy, colon cancer, narcolepsy, depression, obesity, syndrome, angina, kidney disease, cirrhosis, ADHD, osteoarthritis, heart failure, dementia, ailment, stroke, psoriasis, ALS, manic depression, allergy, autism, bronchitis, atherosclerosis, bipolar disorder, lymphoma, ulcer, skin cancer, alcoholism, heart attack, clinical depression, migraine, mental illness, cataract, apnea, sclerosis, eye disease, infertility, heart condition, reflux, asthma attack, SIDS, genetic disease, anxiety disorder, neurological disease, fibrillation, virus, neurological disorder, diarrhea, nearsightedness, anorexia, hay fever, bulimia, pregnancy, degeneration, cholesterol, INJURY, gout, C, back pain, carcinoma, complication, condition, failure

Nq23: Yale Law School, Harvard Business School, Harvard Law School, Brown University, Harvard, City College, Harvard University, Morehouse College, Amherst College, Yale university, Yale, New York University, Columbia University, University of Chicago, University of Wisconsin, Stanford University, Ohio State University, Boston University, University of Southern California, MIT, University of Virginia, Massachusetts Institute of Technology, Northwestern University, University of Michigan, Georgetown University, University of Texas, Syracuse University, Michigan State University, Princeton, law school, George Washington University, University of Georgia, Williams College, Rutgers University, London School of Economics, Emory University, Indiana University, Cornell University, American University, Hunter College, Queens

College, Princeton University, University of Minnesota, Smith College, University of Kansas, University of Washington, Cambridge University, Dartmouth College, Tufts University, Louisiana State University, University of North Carolina, Southern Methodist University, Harvard College, University of Maryland, Oxford University, Howard University, Northeastern University, school of law, San Francisco State University, Duke University, California State University, University of Arizona, Cornell, Juilliard School, Vanderbilt University, University of New Hampshire, University of Toronto, University of Massachusetts, Rice University, State University of New York, University of Missouri, University of Utah, Juilliard, Arizona State University, Columbia, Oxford, Iowa State University, Georgia State University, University of New Mexico, University of Tennessee, Brandeis University, Vassar, Dartmouth, University of Florida, University of Alabama, Trinity College, Texas Christian University, University of Dayton, academy, art school, University of Miami, Pennsylvania State University, Barnard College, Wellesley College, University of Connecticut, Brigham Young University, UC Berkeley, University of Oklahoma, NYU, Carnegie Mellon University, Fordham University, Culinary Institute of America, air force academy, University of Wyoming, U.S. Naval Academy, naval academy, Baylor University, University of Notre Dame, University of Oregon, City University of New York, Hebrew University, University of Mississippi, school of music, University of Nebraska, Temple University, Kansas State University, Georgetown, University of Arkansas, Eton, University of South Carolina, Ohio University, CAMBRIDGE, Catholic University, Tehran University, Florida State University, West Point, Hofstra University, UT, Brandeis, North Carolina State University, University of Maine, Wharton School, University of Cincinnati, Florida International University, Berkeley, Seton Hall University

Nq29: Conchita Martinez, Arantxa Sanchez Vicario, Mary Pierce, Nathalie Tauziat, Jana Novotna, Amanda Coetzer, Iva Majoli, Arantxa Sanchez-Vicario, Lindsay Davenport, Anke Huber, Monica Seles, Julie Halard-Decugis, Carlos Moya, Yevgeny Kafelnikov, Sandrine Testud, Irina Spirlea, Felix Mantilla, Venus Williams, Martina Hingis, Henrieta Nagyova, Dominique Van Roost, Sarah Pitkowski, Steffi Graf, Richard Krajicek, Nathalie Dechy, Goran Ivanisevic, Barbara Paulus, Francisco Clavet, Patty Schnyder, Karol Kucera, Wayne Ferreira, Alex Corretja, Tim Henman, Alberto Berasategui, NATASHA ZVEREVA, American Lindsay Davenport, Cedric Pioline, Gustavo Kuerten, Jerome Golmard, Andre Agassi, Brenda Schultz-McCarthy, Patrick Rafter, Marcelo Rios, Albert Costa, Jonas Bjorkman, Sergi Bruguera, Ai Sugiyama, Thomas Enqvist, Chanda Rubin, Magdalena Maleeva, Michael Chang, Carlos Costa, Amy Frazier, Thomas Muster, Petr Korda, Serena Williams, Lisa Raymond, Ruxandra Dragomir, Magnus Norman, Alexandra Fusai, Marc Rosset, Amelie Mauresmo, Arnaud Boetsch, American Todd Martin, Sabine Appelmans, American Michael Chang, Slava Dosedel, Greg Rusedski, Hernan Gumy, Anna Kournikova, Fabrice Santoro, Nicolas Kiefer, Tommy Haas, Mary Joe Fernandez, Anne-Gaelle Sidot, Karina Habsudova, Bohdan Ulihrach, Magui Serna, Elena Likhovtseva, Byron Black, Andrea Gaudenzi, Magnus Gustafsson, Gabriela Sabatini, Barbara Schett, Alberto Costa, Thomas Johansson, Tamarine Tanasugarn, Malivai Washington, Guillaume Raoux, Dominik Hrbaty, Andrei Medvedev, Silvia Farina, Mariano Puerta, Sebastien Grosjean, Galo Blanco, Yayuk Basuki, Martin Damm, Pete Sampras, Marat Safin, Vincent Spadea, Russian Yevgeny Kafelnikov, Karim Alami, Helena Sukova, ROGER FEDERER, Corina Morariu, Nicolas Escude, Jiri Novak, Nicolas Lapentti, Kimiko Date, Guy Forget, Jacco Eltingh, Arnaud Clement, Peter Nicol, Daniel Vacek, Julian Alonso, Jennifer Capriati, Paul Haarhuis, Todd Martin, American Pete Sampras, Mariano Zabaleta, Mahesh Bhupathi, Brett Steven, Florencia Labat, David Prinosil, Rodney Eyles, Javier Sanchez, Kimberly Po, Gianluca Pozzi, Sargis Sargsian, Michael Stich, Filip Dewulf, Michelle Martin, Magnus Larsson, Jansher Khan, Jeff Tarango, Grant Stafford, Elena Dementieva, Daniel Nestor, Hicham Arazi, Davide Sanguinetti, Mikael Tillstrom, Zhang Ning, American Andre Agassi, Kim Clijsters, Vince Spadea, Mirjana Lucic, John Higgins, Justin Gimelstob, Poul-Erik Hoyer-Larsen, seed, Kristie Boogert

Nq30: manslaughter, racketeering, grand larceny, burglary, sexual assault, larceny, ENDANGERMENT, theft, robbery, mail fraud, fraud, assault, extortion, homicide, conspiracy, crime, bribery, embezzlement, armed robbery, sodomy, securities fraud, forgery, assault and battery, tax evasion, treason, money laundering, corruption, perjury, breach of trust, child abuse, offense, possession, violation, genocide, tampering, shoplifting, espionage, bribe-taking, count, indecency, insider trading, sedition, terrorism, DWI, obstruction, stolen property, contempt of court, violence, battery, spying, carjackings, subversion, fraud charge, solicitation, Blackmail, mischief, murder charge, graft, bombing, enrichment

Nq39: shirt, jacket, sweater, pant, polo shirt, windbreaker, sweatshirt, trouser, pullover, blouse, T-shirt, jean, turtleneck, Cardigan, sweat pants, blue jean, skirt, leather jacket, overcoat, sneaker, tunic, dress shirt, pantsuit, legging, miniskirts, coat, waistcoat, poncho, hat, undershirt, raincoat, tights, dark glasses, loafer, parka, bellbottoms, hot pants, suit, dress, leotards, sandal, bandanna, mitten, camisole, knickers, bandana, baseball cap, tennis shoe, corduroy, mink coat, pants suit, evening gown, rubber boot, cocktail dress, coverall, bodice, shoe, smock, sarong, cowboy hat, nightgown, trench coat, suspender, miniskirt, cloth, bodysuit, bustier, headband, moccasin, khaki, straw hat, cowboy boot, capris, evening dress, fur coat, caftan, evening bag, bathrobe, bow tie, sombrero, necktie, hosiery, combat boot, wristband, ski mask, sleeve, frock, kimono, halter, slicker, sari, kilt, running shoe, ball gown, TOP, chino, brassiere, tote bag, tux, sheath, lapel, petticoat, cap, G-string, wedding dress, waistband, SHORT, bloomers, clothes, wedding gown, slipcover, Stetson, clothing, name tag, handkerchief, belt buckle, shoulder pad, breech, duster, body armor, neckline, cleat, spacesuit, life jacket, toupee, drawstring, pocket watch, finery, bulletproof vest, sportswear, black tie, docker, lingerie, stiletto, wader, Bolero, getup, headgear, face mask

Nq46: Lennox Lewis, George Foreman, Evander Holyfield, Michael Moorer, Riddick Bowe, Mike Tyson, Larry Holmes, Oscar De La Hoya, Oliver McCall, Buster Douglas, Tyson, Holyfield, Francois Botha, Felix Trinidad, Lewis, Julio Cesar Chavez, Ike Quartey, Sugar Ray Leonard, De La Hoya, Roy Jones Jr., Axel Schulz, Pernell Whitaker, Arturo Gatti, Henry Akinwande, Muhammad Ali, Shannon Briggs, Trinidad, Orlin Norris, Joe Frazier, Lou Savarese, Rocky Marciano, Vaughn Bean, Bowe, Roberto Duran, Shane Mosley, Joe Louis, David Reid, Quartey, Sonny Liston, Fernando Vargas, Oba Carr, Roy Jones, Moorer, Andrew Golota, Johnny Tapia, Akinwande, Botha, Michael Grant, Tszyu, Tua, Ali, Jack Dempsey

Nq70: clarinet, flute, saxophone, violin, guitar, oboe, banjo, Cello, trombone, harmonica, piano, mandolin, trumpet, drum, harp, electric guitar, tuba, INSTRUMENT, acoustic guitar, sax, percussion, woodwind, keyboard, viola, bagpipe, accordion, Bass, steel guitar, fiddle, cymbal

Nq162: Indian Ocean, Red Sea, Mediterranean, South China Sea, Aegean Sea, gulf, Arabian Sea, Sea of Japan, Persian Gulf, Aegean, Pacific, Atlantic, Adriatic, Black Sea, Baltic Sea, Mediterranean Sea, Caribbean, Pacific Ocean, Adriatic Sea, Gulf of Mexico, Caspian Sea, Bay of Bengal, East China Sea, sea, South Atlantic, Yellow Sea, Beibu Gulf, Atlantic Ocean, Bering Sea, Caribbean Sea, North Atlantic, Arabian Peninsula, South Pacific, North Pacific, Baltic, Manila Bay, Western Pacific, strait, Barents Sea, English Channel, North Sea

Nq172: pink, red, turquoise, blue, purple, green, yellow, beige, ORANGE, taupe, color, white, lavender, fuchsia, brown, gray, Black, mauve, royal blue, VIOLET, chartreuse, deep red, teal, dark red, Aqua, gold, burgundy, lilac, crimson, black and white, garnet

Nq202: Antelope, deer, rhino, elephant, giraffe, Leopard, Bengal tiger, tiger, rhinoceros, elk, cheetah, snow leopard, gazelle, wild boar, black bear, zebra, bighorn sheep, bison, lion, hippo, bear, panda, brown bear, wolf, gray wolf, Red Deer, macaque, moose, mastodon, gorilla, blue whale, Gibbon, pheasant, kangaroo, elephant population, golden eagle, sperm whale, bighorn

Nq326: novel, book, memoir, biography, essay, autobiography, poem, short story, novella, magazine article, monograph, film, fiction, picture book, nonfiction, cookbook, tome, literary work, diary, best seller, anthology, Treatise, documentary, writing, movie, bestseller, bibliography, poetry, Good Book, prose, screenplay, newspaper clipping, libretto, manuscript, love letter, paperback, trilogy, life story, glossary, travelogue, yearbook, piece of work, newspaper article, reminiscence, scrapbook, literature, script, reportage, Satanic Verses, LOLITA

Nq474: Manitoba, Saskatchewan, British Columbia, Alberta, Ontario, New Brunswick, Newfoundland, Quebec, Nova Scotia, Prince Edward Island

Nq475: master's degree, bachelor's degree, doctorate, law degree, Ph.D, degree, B.A, MBA, laude, honorary degree, diploma, B.S, knighthood, MASTER

Nq743: orchestra, ensemble, troupe, choir, symphony orchestra, opera company, ballet company, band, sextet, jazz band, chamber orchestra, chorus, bagpiper, dance band, military band, theater company, musical group, festival, cast

Nq744: commotion, brouhaha, hubbub, hullabaloo, fuss, firestorm, excitement, mess, hysteria, ruckus, flap, tempest, scare, fanfare, clamor

Nq745: crocodile, alligator, snake, lizard, spider, scorpion, tarantula, hyena, rattlesnake, gecko, python

Nq746: protein, gene, enzyme, receptor, molecule, antibody, leptin, telomerase, peptide, chemical compound, neurotransmitter, dopamine, antigen, serotonin, hemoglobin, mutation, amino acid, angiogenesis, interferon, nitric oxide, electric current, brain wave, isotope, mitochondria, chromosome, RNA, Y chromosome, fatty acid, antifreeze, Herceptin, taxol

Nq832: fame, stardom, notoriety, prominence, respectability, renown, popularity, celebrity, glory, success, immortality, Recognition, wealth

Nq944: width, length, size, height, diameter, thickness, circumference, depth, surface area, speed, weight, distance

Nq1107: juggler, acrobat, dancer, magician, puppeteer, comedian, Clown, entertainer, belly dancer, movie star, celebrity, comic, photographer, ventriloquist

Nq1367: hearsay, innuendo, misinformation, conjecture, propaganda, gossip

Nq1372: ravine, gully, crevasse, ditch, gorge, CANYON, BAY, crater

Appendix B

1% random sample of the polysemous words discovered by CBC on S_{13403}

argument			
	Nq623	0.34	(tug-of-war, custody battle, wrangling, public debate)
	Nq729	0.18	(assertion, suggestion, notion, contention)
beating			
	Nq712	0.31	(mutilation, cannibalism, strangulation, torture)
	Nq594	0.14	(annihilation, deportation, extermination, persecution)
	Nq636	0.12	(spasm, twinge, contraction)
buoyancy			
	Nq573	0.19	(fluctuation, volatility, gyration, run-up)
	Nq628	0.16	(cohesion, coherence, continuity, predictability)
challenger			
	Nq181	0.25	(candidate, nominee, opponent, contender)
	Nq564	0.11	(shuttle, spacecraft, spaceship, discovery)
communication			
	Nq335	0.32	(avionics, electronic warfare, sonar, propulsion)
	Nq798	0.20	(teamwork, coordination, interaction, bonding)
	Nq442	0.19	(city planning, public works, forestry, social welfare)
	Nq364	0.13	(telex, fax, electronic mail, facsimile)
	Nq306	0.11	(logging, drilling, mining)
cost cutting			
	Nq726	0.30	(damage control, stabilization, cost cutting, eradication)
	Nq107	0.13	(revenue, earnings, profit, income)
delegate			
	Nq541	0.25	(executive council, standing committee, general assembly, legislative assembly)
	Nq61	0.18	(legislator, lawmaker, leader, official)
domination			
	Nq454	0.30	(supremacy, preeminence, dominance, primacy)
	Nq562	0.19	(imperialism, colonialism, fascism, totalitarianism)
equilibrium			
	Nq771	0.27	(equilibrium, reciprocity, parity, coexistence)
	Nq334	0.11	(tranquility, tranquillity, serenity, stillness)
	Nq536	0.11	(shamble, flux, disarray, limbo)
firmness			
	Nq628	0.15	(cohesion, coherence, continuity, predictability)
	Nq573	0.12	(fluctuation, volatility, gyration, run-up)
gasoline			
	Nq14	0.43	(natural gas, diesel fuel, gasoline, heating oil)
	Nq540	0.11	(cooking oil, dairy product, legume, cereal)
head of state			
	Nq872	0.32	(cabinet minister, finance minister, head of state)
	Nq100	0.11	(organization, group, government, coalition)
imprisonment			
	Nq594	0.23	(annihilation, deportation, extermination, persecution)
	Nq200	0.21	(prison term, penalty, sentence, fine)
judge			
	Nq826	0.37	(military court, tribunal, magistrate)
	Nq418	0.17	(auditor, appraiser, accountant, adjuster)
logic			
	Nq628	0.18	(cohesion, coherence, continuity, predictability)
	Nq827	0.13	(gist, crux, good part, thrust)
	Nq706	0.10	(genetic engineering, biotechnology, artificial intelligence, computer technology)
	Nq209	0.10	(strategy, policy, method, tactic)
military training			
	Nq687	0.19	(vocational training, tutoring, vocational education, retraining)
	Nq158	0.10	(equipment, computer, technology, system)
neutron			
	Nq444	0.41	(neutron, electron, neutrino, proton)
	Nq735	0.11	(ultraviolet light, ultraviolet radiation, magnetic field, radio wave)
pan			
	Nq83	0.41	(skillet, saucepan, frying pan, casserole)
	Nq50	0.11	(machete, ax, hatchet, baseball bat)
plank			
	Nq613	0.14	(armoire, bookcase, dining table, paneling)
	Nq135	0.13	(concrete, brick, marble, tile)
privilege			
	Nq810	0.25	(prerogative, legal right, right to vote, discretion)
	Nq269	0.11	(plaudit, accolade, kudos, commendation)
real estate agent			
	Nq553	0.23	(real estate broker, stockbroker, businesswoman, insurance broker)
	Nq268	0.12	(resident, citizen, voter, homeowner)
robbery			
	Nq817	0.35	(stolen property, mischief, trespass, battery)
	Nq712	0.18	(mutilation, cannibalism, strangulation, torture)

Nq80	0.10	(accident, earthquake, explosion, quake)
sewing		
Nq754	0.15	(carpentry, woodworking, interior design, gardening)
Nq287	0.14	(free time, leeway, extra time, leisure time)
sound system		
Nq304	0.17	(public address system, loudspeaker, megaphone, bullhorn)
Nq158	0.14	(equipment, computer, technology, system)
styrofoam		
Nq397	0.24	(polystyrene, polyethylene, fiberglass, foam)
Nq85	0.11	(container, bottle, bag, jar)
theology		
Nq2	0.32	(mechanical engineering, political science, anthropology, sociology)
Nq209	0.11	(strategy, policy, method, tactic)
tumble		
Nq573	0.12	(fluctuation, volatility, gyration, run-up)
Nq803	0.10	(jaunt, detour, trek, jog)
water		
Nq721	0.37	(liquid, blood, moisture, water)
Nq361	0.20	(ocean, pond, lagoon, creek)
Nq593	0.10	(tide, surf, wave, swell)

Appendix C

1% random sample of the senses discovered by CBC on S_{13403} used for our manual evaluation

- acronym
  - S_1: last name, first name, surname, title
- agenda
  - S_1: guideline, framework, timetable, blueprint
  - S_2: ideology, dogma, orthodoxy, principle
- alternative
  - S_1: option, way, means, solution
- anti-Semitism
  - S_1: racism, bigotry, prejudice, sexism
- aria
  - S_1: song, ballad, folk song, tune
- assist
  - S_1: kiss, hug, embrace, pat
- avant garde
  - S_1: life, archeology, professional life, folklore
- ballet dancer
  - S_1: singer, musician, artist, guitarist
- basketball
  - S_1: football, soccer, baseball, volleyball
- belly
  - S_1: leg, neck, arm, chest
- black bear
  - S_1: deer, rabbit, squirrel, coyote
- bluster
  - S_1: nonsense, lie, fabrication, baloney
- borough
  - S_1: city, country, state, community
- breeze
  - S_1: sea breeze, air mass, gust, weather
  - S_2: shock wave, ripple, tremor, shudder
- bull
  - S_1: torero, bruin, cougar, sea king
  - S_2: horse, elephant, donkey, camel
- cab
  - S_1: car, pickup truck, van, patrol car
- capital
  - S_1: money, donation, funding, honorarium
  - S_2: camp, shantytown, township, slum
- caster
  - S_1: toying, tipster, swinger, stopgap
  - S_2: equipment, test equipment, microcomputer, video equipment
- chalet
  - S_1: home, apartment, duplex, condominium
- chicken broth
  - S_1: chicken stock, soy sauce, broth, tomato sauce
- circumstance
  - S_1: condition, situation, economic condition, financial condition
  - S_2: free time, cash, extra time, credit
  - S_3: identity, whereabouts, motive, presence
- closed session
  - S_1: meeting, session, conference, news conference
- collector
  - S_1: enthusiast, buff, fan, lover
- commuter
  - S_1: motorist, passenger, tourist, traveler
- condor
  - S_1: bald eagle, owl, bird, whooping crane
- contaminant
  - S_1: chemical, pesticide, substance, pollutant
- cornea
  - S_1: kidney, bone marrow, liver, pancreas
- cover charge
  - S_1: fee, property tax, cost, payment
- crossing
  - S_1: train station, railroad station, railway station, hotel
- cynic
  - S_1: critic, proponent, opponent, advocate
- debut
  - S_1: premiere, preview, opening, world premiere
- demonstration
  - S_1: sit-in, protest, work stoppage, protest march
- devotion
  - S_1: commitment, loyalty, dedication, allegiance

- disco
  - S_1: discotheque, nightclub, club, theater
  - S_2: reggae, funk, jazz, blue
- dividing line
  - S_1: border, frontier, boundary, county line
- drawer
  - S_1: freezer, closet, pantry, locker
- duration
  - S_1: extent, magnitude, severity, scope
- elected official
  - S_1: member, lawmaker, legislator, activist
- enchilada
  - S_1: taco, burrito, tamale, sandwich
- eroding
  - S_1: toying, tipster, swinger, stopgap
  - S_2: asset, investment, real property, penny stock
- exertion
  - S_1: toying, tipster, swinger, stopgap
  - S_2: stress, exhaustion, strain, hardship
- facsimile machine
  - S_1: equipment, test equipment, microcomputer, video equipment
- feature film
  - S_1: movie, film, comedy, documentary film
- final period
  - S_1: quarter, first period, first half, overtime
- flag
  - S_1: banner, balloon, streamer, national flag
  - S_2: ancestry, descent, heritage, culture
- foil
  - S_1: plastic wrap, wax paper, aluminum foil, cheesecloth
- foster home
  - S_1: shelter, hospice, center, soup kitchen
- frosting
  - S_1: icing, whipped cream, cream, glaze
- garbage truck
  - S_1: truck, school bus, vehicle, tractor trailer
- girder
  - S_1: wall, door, tile roof, sliding door
- good will
  - S_1: peace, reconciliation, conciliation, mutual understanding
  - S_2: courage, determination, patience, strength
- greenhouse
  - S_1: swimming pool, pool, tennis court, sauna
- gum
  - S_1: candy, chewing gum, popcorn, candy bar
- hard hat
  - S_1: glove, goggles, helmet, face mask
- heat exhaustion
  - S_1: illness, infection, ailment, health problem
- honoree
  - S_1: member, lawmaker, legislator, elected official
  - S_2: award, trophy, honorary degree
- human rights
  - S_1: freedom, civil liberty, human right, civil right
- imaging
  - S_1: equipment, test equipment, microcomputer, video equipment
- indian reservation
  - S_1: property, site, piece of land, timberland
- inquiry
  - S_1: study, audit, medical report, police investigation
- international
  - S_1: mutual fund, fund, municipal bond, certificate of deposit
- jakes
  - S_1: organization, group, government, business organization
- justice
  - S_1: judge, tribunal, military court, magistrate
  - S_2: equilibrium, parity, reciprocity, neutrality
- labor market
  - S_1: market, business, housing industry, stock market
- lawsuit
  - S_1: allegation, case, charge, complaint
- letdown
  - S_1: spasm, contraction, cramp, twinge
  - S_2: need, demand, challenge, financial loss

line
S_1: compliance, accordance, conformity, noncompliance
S_2: track, fairway

log cabin
S_1: house, cottage, bungalow, farmhouse

machinist
S_1: flight attendant, personnel, beautician, ticket agent

manse
S_1: home, apartment, duplex, condominium

matter
S_1: issue, question, topic, agenda item

mental hospital
S_1: psychiatric hospital, mental institution, institution, ward

militant
S_1: guerrilla, rebel, insurgent, extremist

mitigation
S_1: study, audit, medical report, police investigation

mosaic
S_1: pottery, piece, figurine, object

musical
S_1: movie, film, comedy, feature film

neglect
S_1: indifference, apathy, complacency, carelessness
S_2: battery, false imprisonment, possession, assault

noise level
S_1: blood pressure, cholesterol, blood sugar, intake

obscurity
S_1: disarray, chaos, limbo, flux

onlooker
S_1: protester, demonstrator, marcher, student

outdoors
S_1: free time, cash, extra time, credit

paintbrush
S_1: shovel, spade, rake, broom

part
S_1: much, most, portion, square foot
S_2: goods, product, consumer goods, semiconductor

pear
S_1: peach, nectarine, apricot, apple

personal loan
S_1: mortgage, loan, second mortgage, credit card

pigment
S_1: resin, coating, fiber, polymer

playing field
S_1: volleyball court, basketball court, picnic area, playground
S_2: equilibrium, parity, reciprocity, justice

political scientist
S_1: historian, sociologist, expert, economist

potter
S_1: adage, proverb, maxim, aphorism

press agent
S_1: company, retailer, distributor, consulting company

production
S_1: gross national product, output, export, labor force
S_2: transportation, distribution, storage, processing
S_3: exposure, concentration, use, toxicity
S_4: premiere, preview, opening, world premiere

provocation
S_1: sabotage, bribe

pusher
S_1: drug trafficker, trafficker, offender, smuggler

radiologist
S_1: doctor, physician, psychiatrist, psychologist

read
S_1: burn, scratch, cut, turn

redemption
S_1: offering, purchase, repurchase, issuance
S_2: bankruptcy, foreclosure, liquidation, insolvency

religious order
S_1: organization, group, government, business organization

resin
S_1: coating, fiber, polymer, packaging
S_2: coal, charcoal, liquid, wood

rich man
S_1: toying, tipster, swinger, stopgap
S_2: liar, traitor, thief, criminal
S_3: wife, daughter, husband, father

roll call
S_1: meeting, session, conference, news conference

runoff

S_1: election, primary election, balloting, primary

S_2: tap water, well water, seawater, syrup

saturation

S_1: toying, tipster, swinger, stopgap

S_2: need, demand, challenge, financial loss

scourge

S_1: infestation, plague, epidemic, swarm

secretary of state

S_1: commissioner, commander, representative, secretary

S_2: governorship, lieutenant governor, judgeship, presidency

S_3: member, lawmaker, legislator, elected official

serenade

S_1: concerto, quartet, sonata, symphony

sheet

S_1: piece of paper, card, sheet of paper, slip of paper

S_2: robe, handkerchief, bathrobe, shawl

S_3: pipe, tube, tubing, rod

shrinkage

S_1: slowdown, decline, upturn, growth

skateboarding

S_1: skiing, fishing, horseback riding, swimming

slur

S_1: slogan, graffiti, swastika, epithet

software

S_1: operating system, software product, software package, computer program

S_2: copyright, patent, trademark, property right

sparring

S_1: squabble, feud, spat, quarrel

sports section

S_1: newspaper, magazine, tabloid, publication

standby

S_1: respirator, ventilator, support system, auction block

stickler

S_1: contempt, disregard, disdain, disrespect

striker

S_1: protester, demonstrator, marcher, student

suitability

S_1: effectiveness, validity, reliability, feasibility

sweating

S_1: headache, chest pain, nausea, diarrhea

tale

S_1: thriller, love story, fable, saga

technology

S_1: biotechnology, genetic engineering, artificial intelligence, computer technology

S_2: equipment, test equipment, microcomputer, video equipment

test equipment

S_1: equipment, microcomputer, video equipment, technology

tide

S_1: wave, swell, surf, rising tide

torso

S_1: leg, neck, arm, chest

transformation

S_1: transition, evolution, changeover, metamorphosis

trooper

S_1: officer, peace officer, patrolman, prison guard

tweed

S_1: silk, wool, cotton, leather

upper

S_1: top, side, edge, bottom

vendor

S_1: producer, exporter, importer, supplier

visit

S_1: trip, tour, excursion, vacation

warhorse

S_1: filly, colt, mare, gelding

weekly

S_1: newspaper, magazine, tabloid, publication

willingness

S_1: intention, readiness, sincerity, intent

S_2: discipline, professionalism, sportsmanship, awareness

works

S_1: plant, facility, refinery, oil refinery

zone

S_1: camp, shantytown, township, slum